Final Project Submission

Please fill out:

- Student name: Tia Plagata
- Student pace: full time
- Scheduled project review date/time: Wed, Sep 16, 2020, 10:30 AM - 11:00 AM EST
- Instructor name: Rafael Carrasco
- Blog post URL:

Data Importation & Preparation

1) Unzipping Data Files using Patool Library

In [23]:

```
! pip install patool
```

```
Collecting patool
  Downloading patool-1.12-py2.py3-none-any.whl (77 kB)
     |████████████████████████████████| 77 kB 1.1 MB/s eta 0:00:01
Installing collected packages: patool
Successfully installed patool-1.12
```

In [2]:

```
import patoolib
import os
```

In [26]:

```
# Change our directory
os.chdir("/Users/jordanrjohnson/DataScienceCourseMaterial/phase_1/phase_1_pr
```

In [28]:

```
# Check if our zipped files are in this directory
os.listdir()
```

Out[28]:

```
['imdb.title.crew.csv.gz',
 '.DS_Store',
 'tmdb.movies.csv.gz',
 'upzipped data',
 'imdb.title.akas.csv.gz',
 'imdb.title.ratings.csv.gz',
 'imdb.name.basics.csv.gz',
 'rt.reviews.tsv.gz',
 'imdb.title.basics.csv.gz',
 'rt.movie_info.tsv.gz',
 'tn.movie_budgets.csv.gz',
 'bom.movie_gross.csv.gz',
 'imdb.title.principals.csv.gz']
```

In [30]:

```
# Unzip files
patoolib.extract_archive('imdb.title.crew.csv.gz')
```

```
patool: Extracting imdb.title.crew.csv.gz ...
patool: running '/usr/bin/gzip' -c -d -- 'imdb.title.crew.csv.gz' >
'./Unpack_3r3gnlpw/imdb.title.crew.csv'
patool:     with shell='True'
patool: ... imdb.title.crew.csv.gz extracted to `imdb.title.crew.csv'.
```

Out[30]:

```
'imdb.title.crew.csv'
```

In [36]:

```
patoolib.extract_archive('bom.movie_gross.csv.gz')
```

```
patool: Extracting bom.movie_gross.csv.gz ...
patool: running '/usr/bin/gzip' -c -d -- 'bom.movie_gross.csv.gz' >
'./Unpack_q4h9pv75/bom.movie_gross.csv'
patool:     with shell='True'
patool: ... bom.movie_gross.csv.gz extracted to `bom.movie_gross.csv'.
```

Out[36]:

```
'bom.movie_gross.csv'
```

In [40]:

```
patoolib.extract_archive('imdb.title.basics.csv.gz')
```

```
patool: Extracting imdb.title.basics.csv.gz ...
patool: running '/usr/bin/gzip' -c -d -- 'imdb.title.basics.csv.gz' >
'./Unpack_2h_25kjr/imdb.title.basics.csv'
patool:     with shell='True'
patool: ... imdb.title.basics.csv.gz extracted to `imdb.title.basics.c
sv'.
```

Out[40]:

```
'imdb.title.basics.csv'
```

In [44]:

```
patoolib.extract_archive('imdb.title.ratings.csv.gz')
```

```
patool: Extracting imdb.title.ratings.csv.gz ...
patool: running '/usr/bin/gzip' -c -d -- 'imdb.title.ratings.csv.gz' >
'./Unpack_bewuh5_j/imdb.title.ratings.csv'
patool:     with shell='True'
patool: ... imdb.title.ratings.csv.gz extracted to `imdb.title.rating
s.csv'.
```

Out[44]:

```
'imdb.title.ratings.csv'
```

In [48]:

```
patoolib.extract_archive('tn.movie_budgets.csv.gz')
```

```
patool: Extracting tn.movie_budgets.csv.gz ...
patool: running '/usr/bin/gzip' -c -d -- 'tn.movie_budgets.csv.gz' >
'./Unpack_g8rk87_u/tn.movie_budgets.csv'
patool:     with shell='True'
patool: ... tn.movie_budgets.csv.gz extracted to `tn.movie_budgets.cs
v'.
```

Out[48]:

```
'tn.movie_budgets.csv'
```

In [49]:

```
# Check if unzipped files are in our directory
os.listdir()
```

Out[49]:

```
['imdb.title.crew.csv.gz',
 '.DS_Store',
 'tmdb.movies.csv.gz',
 'imdb.title.crew.csv',
 'upzipped data',
 'tn.movie_budgets.csv',
 'imdb.title.ratings.csv',
 'imdb.title.akas.csv.gz',
 'imdb.title.ratings.csv.gz',
 'imdb.name.basics.csv.gz',
 'bom.movie_gross.csv',
 'imdb.title.basics.csv',
 'rt.reviews.tsv.gz',
 'imdb.title.basics.csv.gz',
 'rt.movie_info.tsv.gz',
 'tn.movie_budgets.csv.gz',
 'bom.movie_gross.csv.gz',
 'imdb.title.principals.csv.gz']
```

2) Creating Pandas DataFrames to explore tables

In [3]:

```
import pandas as pd
import numpy as np
```

In [4]:

```
# Open and read these files as pandas dataframes
bom_gross = pd.read_csv('/Users/jordanrjohnson/DataScienceCourseMaterial/pha
```

In [5]:

```
title_basics = pd.read_csv('/Users/jordanrjohnson/DataScienceCourseMaterial/
```

In [6]:

```
title_ratings = pd.read_csv('/Users/jordanrjohnson/DataScienceCourseMaterial
```

In [7]:

```
movie_budgets = pd.read_csv('/Users/jordanrjohnson/DataScienceCourseMaterial
```

In [8]:

```
movie_budgets.head()
```

Out[8]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
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

In [9]:

```
# Explore the movie_budgets DF
movie_budgets['release_date'].unique
```

Out[9]:

```
<bound method Series.unique of 0       Dec 18, 2009
1       May 20, 2011
2        Jun 7, 2019
3        May 1, 2015
4       Dec 15, 2017
           ...
5777    Dec 31, 2018
5778     Apr 2, 1999
5779    Jul 13, 2005
5780    Sep 29, 2015
5781     Aug 5, 2005
Name: release_date, Length: 5782, dtype: object>
```

In [10]:

```
movie_budgets.shape
```

Out[10]:

```
(5782, 6)
```

In [11]:

```
# Explore the bom_gross DF
bom_gross.head()
```

Out[11]:

	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 [12]:

```
bom_gross.shape
```

Out[12]:

```
(3387, 5)
```

In [13]:

```
bom_gross['studio'].unique()
```

Out[13]:

```
array(['BV', 'WB', 'P/DW', 'Sum.', 'Par.', 'Uni.', 'Fox', 'Wein.', 'So
ny',
       'FoxS', 'SGem', 'WB (NL)', 'LGF', 'MBox', 'CL', 'W/Dim.', 'CB
S',
       'Focus', 'MGM', 'Over.', 'Mira.', 'IFC', 'CJ', 'NM', 'SPC', 'Pa
rV',
       'Gold.', 'JS', 'RAtt.', 'Magn.', 'Free', '3D', 'UTV', 'Rela.',
       'Zeit.', 'Anch.', 'PDA', 'Lorb.', 'App.', 'Drft.', 'Osci.', 'I
W',
       'Rog.', nan, 'Eros', 'Relbig.', 'Viv.', 'Hann.', 'Strand', 'NG
E',
       'Scre.', 'Kino', 'Abr.', 'CZ', 'ATO', 'First', 'GK', 'FInd.',
       'NFC', 'TFC', 'Pala.', 'Imag.', 'NAV', 'Arth.', 'CLS', 'Mont.',
       'Olive', 'CGld', 'FOAK', 'IVP', 'Yash', 'ICir', 'FM', 'Vita.',
       'WOW', 'Truly', 'Indic.', 'FD', 'Vari.', 'TriS', 'ORF', 'IM',
       'Elev.', 'Cohen', 'NeoC', 'Jan.', 'MNE', 'Trib.', 'Rocket',
       'OMNI/FSR', 'KKM', 'Argo.', 'SMod', 'Libre', 'FRun', 'WHE', 'P
4',
       'KC', 'SD', 'AM', 'MPFT', 'Icar.', 'AGF', 'A23', 'Da.', 'NYer',
       'Rialto', 'DF', 'KL', 'ALP', 'LG/S', 'WGUSA', 'MPI', 'RTWC', 'F
IP',
       'RF', 'ArcEnt', 'PalUni', 'EpicPics', 'EOne', 'LD', 'AF', 'TF
A',
       'Myr.', 'BM&DH', 'SEG', 'PalT', 'Outs', 'OutF', 'BSM', 'WAMCR',
       'PM&E', 'A24', 'Cdgm.', 'Distrib.', 'Imax', 'PH', 'HTR', 'ELS',
       'PI', 'E1', 'TVC', 'FEF', 'EXCL', 'MSF', 'P/108', 'FCW', 'XL',
       'Shout!', 'SV', 'CE', 'VPD', 'KE', 'Saban', 'CF&SR', 'Triu', 'D
R',
       'Crnth', 'Ampl.', 'CP', 'Proud', 'BGP', 'Abk.', 'DLA', 'B360',
       'BWP', 'SEA', 'RME', 'KS', 'VE', 'LGP', 'EC', 'FUN', 'STX', 'A
R',
       'BG', 'PFR', 'BST', 'BH Tilt', 'BSC', 'U/P', 'UHE', 'CLF', 'F
R',
       'AaF', 'Orch.', 'Alc', 'PBS', 'SHO', 'Grav.', 'Gathr', 'Asp.',
       'ADC', 'Rel.', 'SM', 'AZ', 'UEP', 'ITL', 'TA', 'MR', 'BBC',
       'CFilms', 'Part.', 'FOR', 'TAFC', 'JBG', 'PNT', 'CineGalaxy',
       'Fathom', 'Zee', 'Men.', 'YFG', 'Gaatri', 'Mon', 'Ghop',
       'Cleopatra', 'Dreamwest', 'SDS', 'Linn', 'Electric', 'Jampa',
'HC',
       'GrtIndia', 'Neon', 'ENTMP', 'Good Deed', 'ParC', 'Aviron',
       'Annapurna', 'Amazon', 'Affirm', 'MOM', 'Orion', 'CFI', 'UTMW',
       'Crimson', 'CAVU', 'EF', 'Arrow', 'Hiber', 'Studio 8',
       'Global Road', 'Trafalgar', 'Greenwich', 'Spanglish', 'Blue Fo
x',
       'RLJ', 'Swen', 'PackYourBag', 'Gaum.', 'Grindstone',
       'Conglomerate', 'MUBI', 'Darin Southa', 'Super', 'CARUSEL', 'PD
F',
       'Synergetic'], dtype=object)
```

In [14]:

```
bom_gross['year'].unique()
```

Out[14]:

```
array([2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018])
```

In [15]:

```
# Explore title_basics DF
title_basics.head()
```

Out[15]:

	tconst	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

In [16]:

```
title_basics['start_year'].unique()
```

Out[16]:

```
array([2013, 2019, 2018, 2017, 2012, 2010, 2011, 2015, 2021, 2016, 201
4,
       2020, 2022, 2023, 2024, 2026, 2025, 2115, 2027])
```

Question 1: What are the qualities of the highest grossing films?

Which studios are producing the highest domestic grossing films?

Clean/prepare the table(s)

- Get rid of null values
- Get rid of duplicates

Note: Since we are just starting our movie production business, let's just focus on domestic gross for n

In [17]:

```
# Check for nulls in domestic_gross column
# Most are foreign titles and there are not too many of them
bom_gross.loc[bom_gross['domestic_gross'].isna() == True]
```

Out[17]:

	title	studio	domestic_gross	foreign_gross	year
230	It's a Wonderful Afterlife	UTV	NaN	1300000	2010
298	Celine: Through the Eyes of the World	Sony	NaN	119000	2010
302	White Lion	Scre.	NaN	99600	2010
306	Badmaash Company	Yash	NaN	64400	2010
327	Aashayein (Wishes)	Relbig.	NaN	3800	2010
537	Force	FoxS	NaN	4800000	2011
713	Empire of Silver	NeoC	NaN	19000	2011
871	Solomon Kane	RTWC	NaN	19600000	2012
928	The Tall Man	Imag.	NaN	5200000	2012
933	Keith Lemon: The Film	NaN	NaN	4000000	2012

In [18]:

```
# Check for nulls in studio column
# We can drop all these as well since there are only 5 of them
bom_gross.loc[bom_gross['studio'].isna() == True]
```

Out[18]:

	title	studio	domestic_gross	foreign_gross	year
210	Outside the Law (Hors-la-loi)	NaN	96900.0	3300000	2010
555	Fireflies in the Garden	NaN	70600.0	3300000	2011
933	Keith Lemon: The Film	NaN	NaN	4000000	2012
1862	Plot for Peace	NaN	7100.0	NaN	2014
2825	Secret Superstar	NaN	NaN	122000000	2017

In [19]:

```
# Drop nulls
bom_gross = bom_gross.dropna(subset = ['domestic_gross', 'studio'])
```

In [20]:

```
# Check for nulls in DF
bom_gross.isna().sum()
```

Out[20]:

```
title                0
studio               0
domestic_gross       0
foreign_gross     1349
year                 0
dtype: int64
```

In [91]:

```
# Look at nulls for foreign gross -- there are alot of them
bom_gross.loc[bom_gross['foreign_gross'].isna() == True]
```

Out[91]:

	title	studio	domestic_gross	foreign_gross	year
222	Flipped	WB	1800000.0	NaN	2010
254	The Polar Express (IMAX re-issue 2010)	WB	673000.0	NaN	2010
267	Tiny Furniture	IFC	392000.0	NaN	2010
269	Grease (Sing-a-Long re-issue)	Par.	366000.0	NaN	2010
280	Last Train Home	Zeit.	288000.0	NaN	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

1349 rows × 5 columns

In [21]:

```
# Instead of deleting lots of values, let's just drop the column since we ar
bom_gross = bom_gross.drop(columns = 'foreign_gross')
```

In [22]:

```
# Check again for nulls
bom_gross.isna().sum()
```

Out[22]:

```
title             0
studio            0
domestic_gross    0
year              0
dtype: int64
```

In [23]:

```
# Check for duplicates
duplicates = bom_gross[bom_gross.duplicated()]
print(len(duplicates))
```

```
0
```

In [24]:

```
# Check for placeholder values (maybe a negative value)
# Even though values like 1100000 come up many times,
# it is likely valid since many of these values are estimated and rounded
bom_gross['domestic_gross'].value_counts()
```

Out[24]:

```
1100000.0      32
1000000.0      30
1300000.0      30
1200000.0      25
1400000.0      23
               ..
68800.0         1
87000000.0      1
739000.0        1
336000000.0     1
727000.0        1
Name: domestic_gross, Length: 1794, dtype: int64
```

In [25]:

```
# Make sure years are within our desired range
bom_gross['year'].value_counts()
```

Out[25]:

```
2015    449
2016    433
2011    396
2012    393
2014    390
2013    345
2010    322
2017    320
2018    308
Name: year, dtype: int64
```

Grouping and plotting the results

In [26]:

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.style.use('seaborn-white')
```

In [27]:

```
# Group by studio and show us the total each studio has made
bom_studios = bom_gross.groupby('studio')['domestic_gross'].sum().sort_value
bom_studios.head()
```

Out[27]:

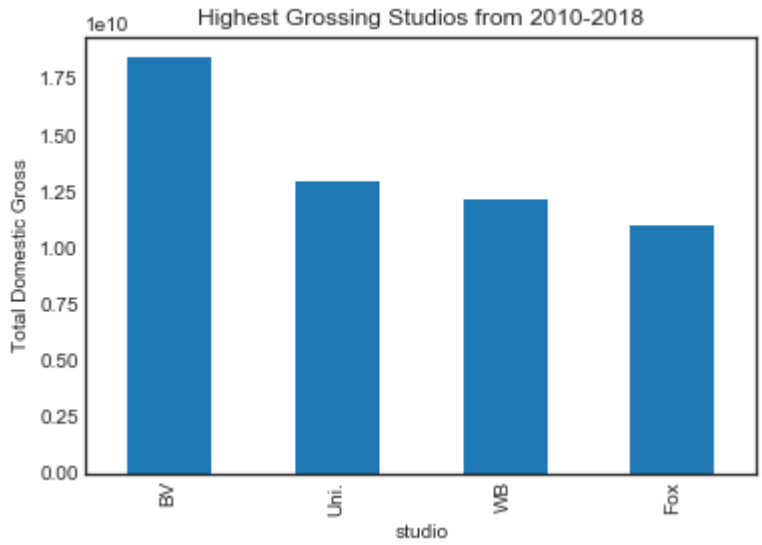
```
studio
BV      1.841903e+10
Uni.    1.290239e+10
WB      1.216805e+10
Fox     1.094950e+10
Sony    8.459683e+09
Name: domestic_gross, dtype: float64
```

In [28]:

```
# Create plot, add title, add ylabel
studios_plt = bom_studios.head(4).plot(kind = 'bar')
plt.title('Highest Grossing Studios from 2010-2018')
plt.ylabel('Total Domestic Gross')
```

Out[28]:

```
Text(0, 0.5, 'Total Domestic Gross')
```



Evaluate further: Which are the highest grossing films that the studios are making?

Let's sort these films by domestic_gross, then find the top 10 films per studio for the top 4 studi

Note: This dataset gives us films from 2010-2018

In [29]:

```
# Sort films by highest gross
sorted_gross = bom_gross.sort_values(ascending = False, by = 'domestic_gross
```

In [30]:

```
# Create DF with just the highest grossing films for Buena Vista
top_BV = sorted_gross.loc[sorted_gross['studio'] == 'BV']
top_BV.head(10)
```

Out[30]:

	title	studio	domestic_gross	year
1872	Star Wars: The Force Awakens	BV	936700000.0	2015
3080	Black Panther	BV	700100000.0	2018
3079	Avengers: Infinity War	BV	678800000.0	2018
727	Marvel's The Avengers	BV	623400000.0	2012
2758	Star Wars: The Last Jedi	BV	620200000.0	2017
3082	Incredibles 2	BV	608600000.0	2018
2323	Rogue One: A Star Wars Story	BV	532200000.0	2016
2759	Beauty and the Beast (2017)	BV	504000000.0	2017
2324	Finding Dory	BV	486300000.0	2016
1875	Avengers: Age of Ultron	BV	459000000.0	2015

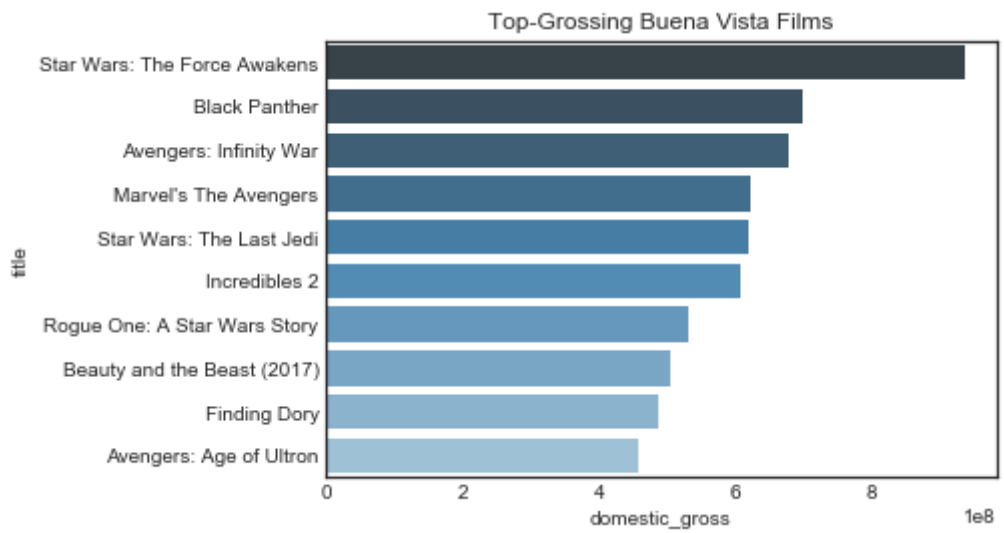
In [31]:

```
# Plot top grossing films for BV
BV_plot = sns.barplot(data = top_BV.head(10),
                      x = 'domestic_gross',
                      y = 'title',
                     palette = 'Blues_d')

BV_plot.set_title('Top-Grossing Buena Vista Films')
BV_plot
```

Out[31]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2a1d8ef0>
```



In [32]:

```
# Create DF with just the highest grossing films for Universal
top_Uni = sorted_gross.loc[sorted_gross['studio'] == 'Uni.']
top_Uni.head(10)
```

Out[32]:

	title	studio	domestic_gross	year
1873	Jurassic World	Uni.	652300000.0	2015
3081	Jurassic World: Fallen Kingdom	Uni.	417700000.0	2018
2327	The Secret Life of Pets	Uni.	368400000.0	2016
1129	Despicable Me 2	Uni.	368100000.0	2013
1874	Furious 7	Uni.	353000000.0	2015
1876	Minions	Uni.	336000000.0	2015
3096	Dr. Seuss' The Grinch (2018)	Uni.	270600000.0	2018
2334	Sing	Uni.	270400000.0	2016
2761	Despicable Me 3	Uni.	264600000.0	2017
8	Despicable Me	Uni.	251500000.0	2010

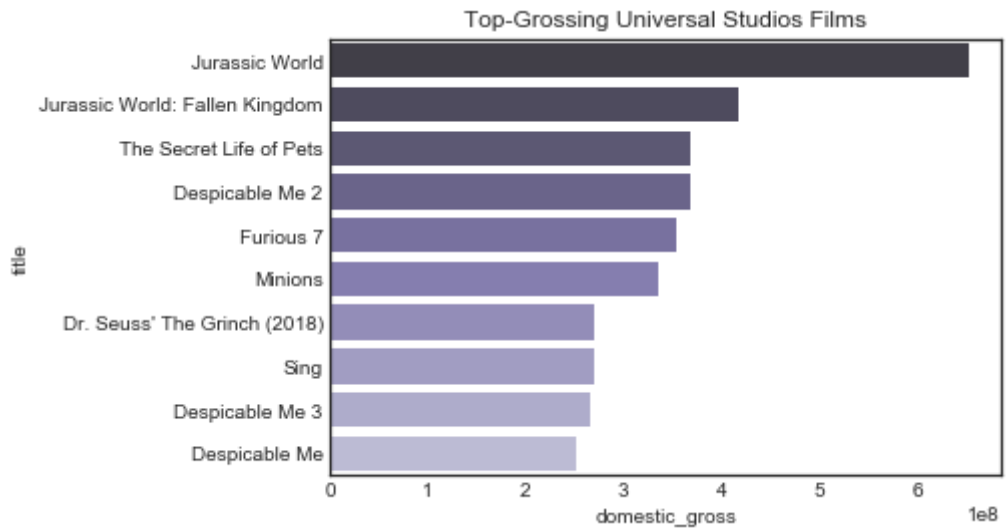
In [33]:

```
# Plot top grossing films for Univeral
Uni_plot = sns.barplot(data = top_Uni.head(10),
                       x = 'domestic_gross',
                       y = 'title',
                      palette = 'Purples_d')

Uni_plot.set_title('Top-Grossing Universal Studios Films')
Uni_plot
```

Out[33]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2a265f60>
```



In [34]:

```
# Create DF with the highest grossing films for Warner Bros
top_WB = sorted_gross.loc[sorted_gross['studio'] == 'WB']
top_WB.head(10)
```

Out[34]:

	title	studio	domestic_gross	year
729	The Dark Knight Rises	WB	448100000.0	2012
2767	Wonder Woman	WB	412600000.0	2017
328	Harry Potter and the Deathly Hallows Part 2	WB	381000000.0	2011
1489	American Sniper	WB	350100000.0	2014
3083	Aquaman	WB	335100000.0	2018
2328	Batman v Superman: Dawn of Justice	WB	330400000.0	2016
2331	Suicide Squad	WB	325100000.0	2016
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	2010
3	Inception	WB	292600000.0	2010
1135	Man of Steel	WB	291000000.0	2013

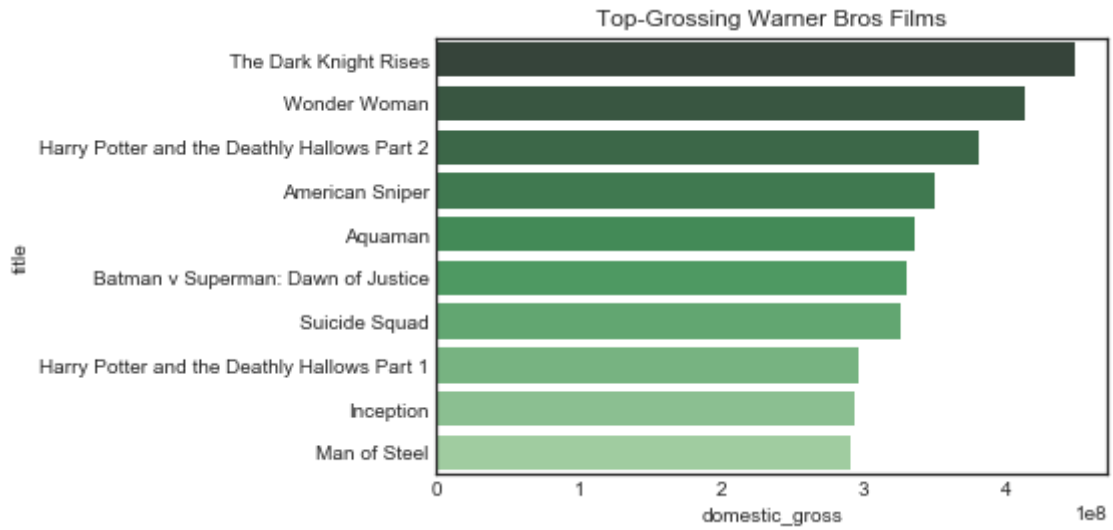
In [35]:

```
# Plot top grossing films for WB
WB_plot = sns.barplot(data = top_WB.head(10),
                      x = 'domestic_gross',
                      y = 'title',
                     palette = 'Greens_d')

WB_plot.set_title('Top-Grossing Warner Bros Films')
WB_plot
```

Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2a4199b0>
```



In [36]:

```
# Create DF with the highest grossing films for Fox
top_Fox = sorted_gross.loc[sorted_gross['studio'] == 'Fox']
top_Fox.head(10)
```

Out[36]:

	title	studio	domestic_gross	year
2330	Deadpool	Fox	363100000.0	2016
3087	Deadpool 2	Fox	318500000.0	2018
1482	X-Men: Days of Future Past	Fox	233900000.0	2014
1881	The Martian	Fox	228400000.0	2015
2772	Logan (2017)	Fox	226300000.0	2017
3084	Bohemian Rhapsody	Fox	216400000.0	2018
1484	Dawn of the Planet of the Apes	Fox	208500000.0	2014
1137	The Croods	Fox	187200000.0	2013
1884	The Revenant	Fox	183600000.0	2015
1890	Home (2015)	Fox	177400000.0	2015

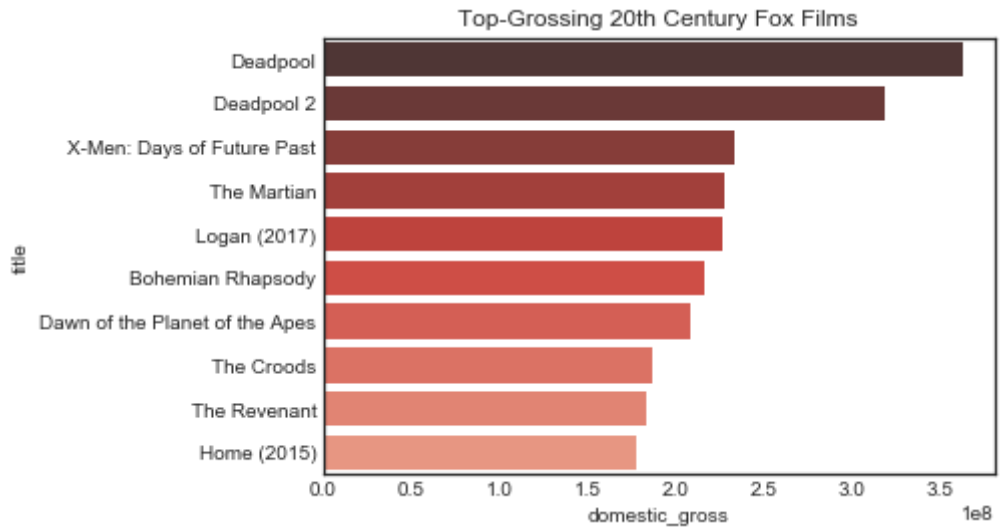
In [37]:

```
# Plot top grossing films for Fox
Fox_plot = sns.barplot(data = top_Fox.head(10),
                       x = 'domestic_gross',
                       y = 'title',
                      palette = 'Reds_d')

Fox_plot.set_title('Top-Grossing 20th Century Fox Films')
Fox_plot
```

Out[37]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2a535668>
```



In [38]:

```
# Maybe dig into this?
amazon = sorted_gross.loc[sorted_gross['studio'] == 'Amazon']
amazon.head(20)
```

Out[38]:

	title	studio	domestic_gross	year
3212	Beautiful Boy (2018)	Amazon	7600000.0	2018
3251	Cold War (2018)	Amazon	4600000.0	2018
3239	Life Itself (2018)	Amazon	4099999.0	2018
3267	You Were Never Really Here	Amazon	2500000.0	2018
3236	Suspiria	Amazon	2500000.0	2018
3257	Don't Worry He Won't Get Far on Foot	Amazon	1400000.0	2018
2902	Wonder Wheel	Amazon	1400000.0	2017

Question 1 Conclusion

What are the the qualities of the top-grossing films from the past 10 years?

To find this conclusion, we can look at the bom_movie_gross dataset from Box Office Mojo. The top-gr films in this case are those with the highest domestic gross. We are looking at domestic gross over fore gross because as a new movie studio, we should conquer our home market before trying to overstretc foreign and domestic markets all at once.

The 4 studios with the top-grossing films are:

- Buena Vista (BV) | total domestic gross: $18.4 billion
- Universal Studios (Uni.) | total domestic gross: $12.9 billion
- Warner Bros. (WB) | total domestic gross: $12.1 billion
- 20th Century Fox (Fox) | total domestic gross: $1.1 billion

We can further investigate these top-grossing films by looking at the top 10 films for each of the top st

(see plots: BV_plot, Uni_plot, WB_plot, and Fox_plot)

We can easily see a pattern in the qualities and genres of these films. Most of them are action films. Th superhero films (e.g. Avengers, Deadpool, The Dark Knight, etc), sci-fi/fantasy franchises (e.g. Star Wa Jurassic World, Harry Potter, Dawn of the Planet of the Apes, etc), and animated films for kids & familie Incredibles, Despicable Me, etc).

Recommendations

Based on these findings, I would recommend making films that reflect these qualities (superhero films, animated films for families, sci-fi/fantasy films, etc.

Investing in a sci-fi/fantasy or superhero franchise seems to be a great investment. We can see a trend films over the past 10 years.

We can also conclude that buying the licensing rights to an popular sci-fi saga/series to adapt into a m would be a good investment as well.

Question 2: Which films made the most money?

What is the relationship between production budget and gros earnings?

Clean/prepare the table(s)

- Check for null values
- Check for duplicates
- Change datatypes as needed
- Limit table to only movies released from 2010 to 2020

In [39]:

```
movie_budgets.head()
```

Out[39]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
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

In [40]:

```
# Check for duplicates
mb_duplicates = movie_budgets[movie_budgets.duplicated()]
print(len(mb_duplicates))
```

```
0
```

In [41]:

```
# Check for null values
movie_budgets.isna().sum()
```

Out[41]:

```
id                    0
release_date          0
movie                 0
production_budget     0
domestic_gross        0
worldwide_gross       0
dtype: int64
```

In [42]:

```
# Check datatypes to make sure we have numbers in gross & budget columns
movie_budgets.dtypes
```

Out[42]:

```
id                     int64
release_date          object
movie                 object
production_budget     object
domestic_gross        object
worldwide_gross       object
dtype: object
```

In [43]:

```
# Let's change this to a datetime object in case we need to use it later
movie_budgets['release_date'] = pd.to_datetime(movie_budgets['release_date']
movie_budgets.head()
```

Out[43]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	2009-12-18	Avatar	$425,000,000	$760,507,625	$2,776,345,279
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875
2	3	2019-06-07	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350
3	4	2015-05-01	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747

In [44]:

```
# Change to an int, removing the commas and $s
movie_budgets['production_budget'] = movie_budgets['production_budget'].str.
```

In [45]:

```
# Change to an int, removing the commas and $s
movie_budgets['domestic_gross'] = movie_budgets['domestic_gross'].str.replac
```

In [46]:

```
# Change to an int, removing the commas and $s
movie_budgets['worldwide_gross'] = movie_budgets['worldwide_gross'].str.repl
```

In [47]:

```
# Check our work
movie_budgets.head()
```

Out[47]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	2009-12-18	Avatar	425000000	760507625	2776345279
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875
2	3	2019-06-07	Dark Phoenix	350000000	42762350	149762350
3	4	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1403013963
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1316721747

In [48]:

```
# Let's only include movies within our 10 year scope
movie_budgets = movie_budgets.loc[(movie_budgets['release_date'] > '2010-01-
                                   (movie_budgets['release_date'] < '2019-12-
```

Sort, plot, and find correlation coefficients

In [49]:

```
# Check out the smallest and largest production budgets
movie_budgets.sort_values(by = 'production_budget')
```

Out[49]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
5780	81	2015-09-29	A Plague So Pleasant	1400	0	0
5777	78	2018-12-31	Red 11	7000	0	0
5772	73	2012-01-13	Newlyweds	9000	4584	4584
5771	72	2015-05-19	Family Motocross	10000	0	0
5760	61	2010-04-02	Breaking Upwards	15000	115592	115592
...	...	...	...	...	...	...
5	6	2015-12-18	Star Wars Ep. VII: The Force Awakens	306000000	936662225	2053311220
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1316721747
3	4	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1403013963
2	3	2019-06-07	Dark Phoenix	350000000	42762350	149762350
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875

2185 rows × 6 columns

In [50]:

```
# Create a really pretty plot
# Use size and hue to make the bigger budgets more obvious
# Set title
# Customize the x and y axes
# Despine to make the plot feel more open and decluttered
g = sns.relplot(
                data = movie_budgets,
                x = "production_budget", y = "domestic_gross",
                hue = "production_budget", size = "production_budget",
                sizes = (10, 200),
                )
g.fig.suptitle('Production Budget & Domestic Gross')
g.set(xscale = "linear", yscale = "linear")
g.ax.xaxis.grid(True, "minor", linewidth = .25)
g.ax.yaxis.grid(True, "minor", linewidth = .25)
g.despine(left = True, bottom = True)
```

Out[50]:

```
<seaborn.axisgrid.FacetGrid at 0x1a2a77e080>
```

In [51]:

```
# Find the correlation coefficient to back up our plot findings with a hard
np.corrcoef(x = movie_budgets['production_budget'],
            y = movie_budgets['domestic_gross'])
```

Out[51]:

```
array([[1.        , 0.73482153],
       [0.73482153, 1.        ]])
```

In [52]:

```
# See if there is more correlation with the worldwide gross than the domesti
np.corrcoef(x = movie_budgets['production_budget'],
            y = movie_budgets['worldwide_gross'])
```

Out[52]:

```
array([[1.        , 0.79646255],
       [0.79646255, 1.        ]])
```

In [53]:

```
# Make the same plot, but use the worldwide gross instead to see the differe
h = sns.relplot(
                data = movie_budgets,
                x = "production_budget", y = "worldwide_gross",
                hue = "production_budget", size = "production_budget",
                sizes = (10, 200),
                )
h.fig.suptitle('Production Budget & Worldwide Gross')
h.set(xscale = "linear", yscale = "linear")
h.ax.xaxis.grid(True, "minor", linewidth = .25)
h.ax.yaxis.grid(True, "minor", linewidth = .25)
h.despine(left = True, bottom = True)
```

Out[53]:

```
<seaborn.axisgrid.FacetGrid at 0x1a2a7303c8>
```

Further Investigation: Find Return On Investment for each film Which films have highest ROI?

Find ROI, sort, and plot

In [54]:

```
movie_budgets.head()
```

Out[54]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875
2	3	2019-06-07	Dark Phoenix	350000000	42762350	149762350
3	4	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1403013963
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1316721747
5	6	2015-12-18	Star Wars Ep. VII: The Force Awakens	306000000	936662225	2053311220

In [56]:

```
# Create a column for return on investment using roi formula
movie_budgets['roi'] = ((movie_budgets['worldwide_gross'] - movie_budgets['p
movie_budgets.head()
```

Out[56]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	roi
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	154.667286
2	3	2019-06-07	Dark Phoenix	350000000	42762350	149762350	-57.210757
3	4	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1403013963	324.384139
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1316721747	315.369636
5	6	2015-12-18	Star Wars Ep. VII: The Force Awakens	306000000	936662225	2053311220	571.016739

In [57]:

```
# Sort these films by highest roi
# Notice the trend in these types of films -- almost all horror
top_rois = movie_budgets.sort_values(by = 'roi', ascending = False).head(20)
top_rois.head(20)
```

Out[57]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	
5679	80	2015-07-10	The Gallows	100000	22764410	41656474	4155
5211	12	2012-01-06	The Devil Inside	1000000	53262945	101759490	1007
5062	63	2011-04-01	Insidious	1500000	54009150	99870886	655
5213	14	2015-04-17	Unfriended	1000000	32789645	64364198	633
4664	65	2010-10-20	Paranormal Activity 2	3000000	84752907	177512032	581
4249	50	2017-01-20	Split	5000000	138141585	278964806	547
5189	90	2014-03-21	Godâ€ s Not Dead	1150000	60755732	63777092	544
4248	49	2017-02-24	Get Out	5000000	176040665	255367951	500
3517	18	2012-05-25	Les Intouchables	10800000	13182281	484873045	438
5063	64	2016-10-21	Moonlight	1500000	27854931	65245512	424
5217	18	2012-05-25	Chernobyl Diaries	1000000	18119640	42411721	414
4250	51	2011-10-21	Paranormal Activity 3	5000000	104028807	207039844	404
4083	84	2014-10-03	Annabelle	6500000	84273813	256862920	385
5014	15	2010-08-27	The Last Exorcism	1800000	41034350	70165900	379
4252	53	2013-09-13	Insidious Chapter 2	5000000	83586447	161921515	313
3755	56	2016-12-21	Dangal	9500000	12391761	294654618	300
5358	59	2011-12-30	Jodaeiye Nader az Simin	800000	7098492	24426169	295
4666	67	2013-06-07	The Purge	3000000	64473115	91266581	294
4254	55	2016-07-22	Lights Out	5000000	67268835	148806510	287
4668	69	2012-10-12	Sinister	3000000	48086903	87727807	282

In [59]:

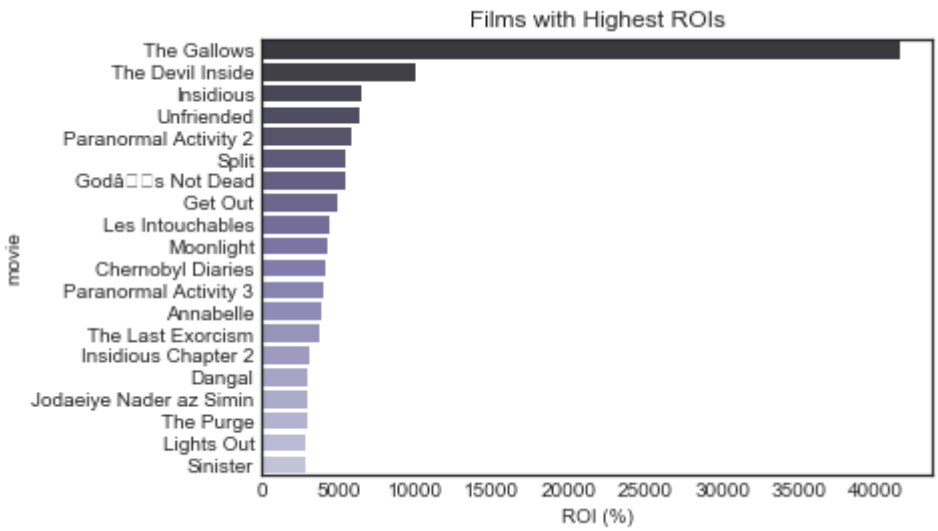
```
# Plot the top 20 films with highest ROIs
top_rois_plot = sns.barplot(data = top_rois,
                            x = 'roi',
                            y = 'movie',
                            palette = 'Purples_d')

top_rois_plot.set_title('Films with Highest ROIs')
top_rois_plot.set_xlabel('ROI (%)')
top_rois_plot
```

Out[59]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2ad80470>
```

```
/Users/jordanrjohnson/opt/anaconda3/envs/learn-env/lib/python3.6/site-
packages/matplotlib/backends/backend_agg.py:211: RuntimeWarning: Glyph
128 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/Users/jordanrjohnson/opt/anaconda3/envs/learn-env/lib/python3.6/site-
packages/matplotlib/backends/backend_agg.py:211: RuntimeWarning: Glyph
153 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/Users/jordanrjohnson/opt/anaconda3/envs/learn-env/lib/python3.6/site-
packages/matplotlib/backends/backend_agg.py:180: RuntimeWarning: Glyph
128 missing from current font.
  font.set_text(s, 0, flags=flags)
/Users/jordanrjohnson/opt/anaconda3/envs/learn-env/lib/python3.6/site-
packages/matplotlib/backends/backend_agg.py:180: RuntimeWarning: Glyph
153 missing from current font.
  font.set_text(s, 0, flags=flags)
```



Question 2 Conclusion

Which films made the most money?

To answer this question, we can look at the tn.movie_budgets database, and narrow down our data se
all films between 2010 and 2020.

The first thing we should ask ourselves is: is there a relationship between the movie's production budg
the movie's gross earnings? When mapped in Seaborn's relplot, we can easily see this relationship.

In "Production Budget and Domestic Gross", we can see that as the production budget increases, so d
domestic gross. We can see that most of the movies with budgets less than 100 million dollars, do not
any more than 500 million dollars in domestic gross earnings. Movies that made over 600 million dollar
production budgets of 200 million dollars or more. Moreover, if we look at the correlation coefficient be
production budget and domestic gross, we can see that there is somewhat strong correlation between
variables (0.73).

In "Production Budget and Worldwide Gross", we can see a similar trend. Our correlation coefficient be
these 2 variables was 0.80, which indicates a stronger correlation than with domestic gross. It is interes
note, that this correlation is not without exception. There were still various films with production budge
between 250 million and 350 million dollars with worldwide gross earnings less than 1 billion dollars.

We can therefore conclude that there is a strong relationship between production budget and gross ea
but what about your return on investment? We can also investigate the type of movies that would prod
highest returns.

Calculating the ROI using the worldwide gross, we can see that many movies lost money (indicated by
negative ROI), but there were also many movies that made 50x or even 400x on their production budge
we look at the top 20 movies with the highest ROIs, we can see right away that most of them are horro

Recommendations

Based on these findings, I would recommend that we should ask ourselves a follow-up question:

- *How big is our production budget?*

If we have a lot to invest, we should take into consideration our findings from question 1, and conclude
can invest in a sci-fi/fantasy/superhero franchise or a film remake, use a large production budget, and
more gross earnings.

However, if we do not want to invest so much, we should make a horror film, in order to receive a large

Question 3: What kind of films get the highest rating

What is the average rating per genre?

Data Cleaning & Tables Preparation

- Join imdb basics and imdb ratings tables

- Check for null values & duplicates
- Make sure data is in correct dtype

In [60]:

```
# Preview the title_basics DF
title_basics.head()
```

Out[60]:

	tconst	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

In [61]:

```
# Preview the title_ratings DF
title_ratings.head()
```

Out[61]:

	tconst	averagerating	numvotes
0	tt10356526	8.3	31
1	tt10384606	8.9	559
2	tt1042974	6.4	20
3	tt1043726	4.2	50352
4	tt1060240	6.5	21

In [62]:

```
# Notice the primary key for both is tconst, so set that as index
title_basics.set_index('tconst', inplace = True)
```

In [63]:

```
title_basics.head()
```

Out[63]:

tconst	primary_title	original_title	start_year	runtime_minutes	genres
tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama
tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama
tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy

In [64]:

```
# Set this DF's index as tconst as well
title_ratings.set_index('tconst', inplace = True)
```

In [65]:

```
title_ratings.head()
```

Out[65]:

tconst	averagerating	numvotes
tt10356526	8.3	31
tt10384606	8.9	559
tt1042974	6.4	20
tt1043726	4.2	50352
tt1060240	6.5	21

In [66]:

```
# Join two DFs on the the tconst key
# Use inner join to only combine the films both DFs have in common
joined_imdb = title_basics.join(title_ratings, how = 'inner')
```

In [67]:

```
# Check out our joined table
# SWEET
joined_imdb.head()
```

Out[67]:

tconst	primary_title	original_title	start_year	runtime_minutes	genres	averag
tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama	
tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama	
tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama	
tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama	
tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	

In [68]:

```
# Check how many films we lost in the join
# RIP, but we need the averagerating AND genres to reach our conclusion
print(title_basics.shape)
print(joined_imdb.shape)
```

```
(146144, 5)
(73856, 7)
```

In [69]:

```
# Investigate this new DF
joined_imdb.describe
```

Out[69]:

```
<bound method NDFrame.describe of                                 primary
_title              original_title  \
tconst
tt0063540                          Sunghursh                   Sunghursh
tt0066787  One Day Before the Rainy Season             Ashad Ka Ek Din
tt0069049       The Other Side of the Wind  The Other Side of the Wind
tt0069204                    Sabse Bada Sukh             Sabse Bada Sukh
tt0100275          The Wandering Soap Opera       La Telenovela Errante
...                                      ...                        ...
tt9913084                  Diabolik sono io            Diabolik sono io
tt9914286                 Sokagin Çocuklari           Sokagin Çocuklari
tt9914642                          Albatross                   Albatross
tt9914942       La vida sense la Sara Amat  La vida sense la Sara Amat
tt9916160                        Drømmeland                  Drømmeland

           start_year  runtime_minutes                genres  averager
ating  \
tconst
tt0063540        2013            175.0    Action,Crime,Drama
7.0
tt0066787        2019            114.0       Biography,Drama
7.2
tt0069049        2018            122.0                 Drama
6.9
tt0069204        2018              NaN          Comedy,Drama
6.1
tt0100275        2017             80.0  Comedy,Drama,Fantasy
6.5
...               ...              ...                   ...
...
tt9913084        2019             75.0           Documentary
6.2
tt9914286        2019             98.0          Drama,Family
8.7
tt9914642        2017              NaN           Documentary
8.5
tt9914942        2019              NaN                   NaN
6.6
tt9916160        2019             72.0           Documentary
6.5

           numvotes
tconst
tt0063540        77
tt0066787        43
tt0069049      4517
tt0069204        13
tt0100275       119
...             ...
tt9913084         6
tt9914286       136
tt9914642         8
tt9914942         5
tt9916160        11
```

```
[73856 rows x 7 columns]>
```

In [70]:

```
# Check for nulls
# Runtime does not affect us for this question, so that's fine
# genres is important though! Let's deal with this in a minute
joined_imdb.isna().sum()
```

Out[70]:

```
primary_title         0
original_title        0
start_year            0
runtime_minutes    7620
genres              804
averagerating         0
numvotes              0
dtype: int64
```

In [71]:

```
# Check for duplicates
joined_duplicates = joined_imdb[joined_imdb.duplicated()]
len(joined_duplicates)
```

Out[71]:

```
0
```

In [72]:

```
# How many unique genres?
joined_imdb['genres'].unique()
len(joined_imdb['genres'].unique())
```

Out[72]:

```
924
```

In [73]:

```
# Instead of deleting data, let's make a category called "no genre" for null
joined_imdb['genres'] = joined_imdb['genres'].fillna(value = "no genre")
```

In [74]:

```
joined_imdb.isna().sum()
```

Out[74]:

```
primary_title         0
original_title        0
start_year            0
runtime_minutes    7620
genres                0
averagerating         0
numvotes              0
dtype: int64
```

In [75]:

```
# Check dtypes to make sure we have numbers where we need them
joined_imdb.dtypes
```

Out[75]:

```
primary_title       object
original_title      object
start_year           int64
runtime_minutes    float64
genres              object
averagerating      float64
numvotes             int64
dtype: object
```

In [76]:

```
# Check to make sure film years are within our 10 year scope
joined_imdb['start_year'].unique()
```

Out[76]:

```
array([2013, 2019, 2018, 2017, 2010, 2011, 2012, 2015, 2016, 2014])
```

Decide the threshold for numvotes

This is a tricky decision to make, because the films with a very small number of votes could REALLY sk
data because the value we have from each averagerating is already skewed. For example, a film with 5
could have 8.0s and up, making it's averagerating very high. Whereas a big box office film will have ov
million reviews, all ranging from 4.0 to 9.0 and its averagerating will be more normalized due to a highe
number of votes.

My approach to this problem was to check out the highest and lowest numvotes, and then plot the nur
to see the distribution. We could take the inner quartile here, but then we are taking a TON of films with
small number of votes.

Instead, I decided to rule out any films with numvotes less than 50,000. To me, 50,000 votes should giv
a pretty normalized averagerating, and we can avoid the films with skewed averageratings.

In [77]:

```
# Check highest and lowest number of votes
joined_imdb['numvotes'].sort_values().head()
joined_imdb['numvotes'].sort_values().tail()
```

Out[77]:

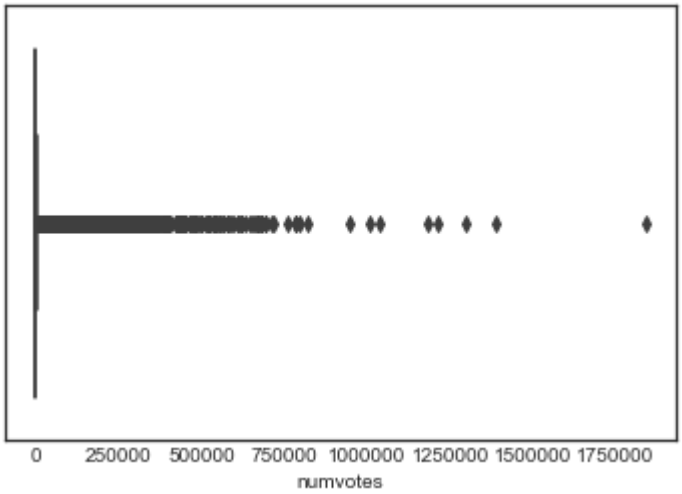
```
tconst
tt0848228    1183655
tt1853728    1211405
tt0816692    1299334
tt1345836    1387769
tt1375666    1841066
Name: numvotes, dtype: int64
```

In [78]:

```
# See distribution of numvotes
# Most of them are close to zero... we don't want these
joined_imdb['numvotes'].sort_values()
sns.boxplot(data=joined_imdb['numvotes'], x=joined_imdb['numvotes'])
```

Out[78]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x123f4f2b0>
```



In [79]:

```
# Let's only include films with over 50,000 votes so that the movies with ve
adjusted_imdb = joined_imdb.loc[joined_imdb['numvotes'] > 50000]
print(adjusted_imdb.shape)
print(joined_imdb.shape)
```

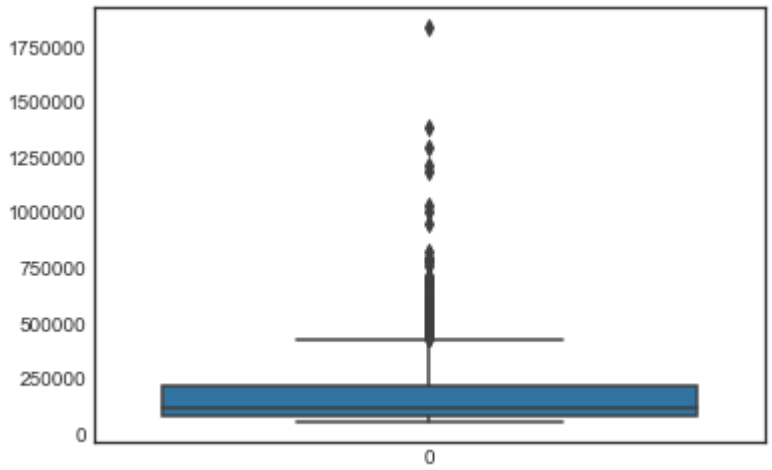
```
(1064, 7)
(73856, 7)
```

In [80]:

```
# Check the distribution again
# This looks much better, and even though some films have an outlier in numv
# their averagerating is still normalized
sns.boxplot(data=adjusted_imdb['numvotes'])
```

Out[80]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x121a5a3c8>
```



Deal with genres column

The next road bump in this DF is dealing with the genres column. Imdb has listed multiple genres for ea film... and they're all in the same column... sigh.

We need to flatten this DF, which will give us duplicated films, but one genre for each value with a corresponding rating. Even though the films' ratings will be counted more than once for different genre interested in finding the ratings per genre so this is still ok. For example, Jurassic World would give a 7 to Action, Adventure and Sci-Fi genres. In the end we will find the median and distribution for each ger counting a film's rating in more than one genre category is still a valid way to find the averagerating pe

In [81]:

```
# Check out our adjusted DF
adjusted_imdb.head()
```

Out[81]:

tconst	primary_title	original_title	start_year	runtime_minutes	genres	a
tt0359950	The Secret Life of Walter Mitty	The Secret Life of Walter Mitty	2013	114.0	Adventure,Comedy,Drama	
tt0365907	A Walk Among the Tombstones	A Walk Among the Tombstones	2014	114.0	Action,Crime,Drama	
tt0369610	Jurassic World	Jurassic World	2015	124.0	Action,Adventure,Sci-Fi	
tt0376136	The Rum Diary	The Rum Diary	2011	119.0	Comedy,Drama	
tt0398286	Tangled	Tangled	2010	100.0	Adventure,Animation,Comedy	

In [82]:

```
# Let's see which movies have the highest ratings after we've adjusted for o
highratings = adjusted_imdb.sort_values(by = 'averagerating', ascending = Fa
highratings.head(20)
```

Out[82]:

tconst	primary_title	original_title	start_year	runtime_minutes	genres	a
tt5813916	The Mountain II	Dag II	2016	135.0	Action,Drama,War	
tt1375666	Inception	Inception	2010	148.0	Action,Adventure,Sci-Fi	
tt4154796	Avengers: Endgame	Avengers: Endgame	2019	181.0	Action,Adventure,Sci-Fi	
tt0816692	Interstellar	Interstellar	2014	169.0	Adventure,Drama,Sci-Fi	
tt1424432	Senna	Senna	2010	106.0	Biography,Documentary,Sport	
tt1675434	The Intouchables	Intouchables	2011	112.0	Biography,Comedy,Drama	
tt2582802	Whiplash	Whiplash	2014	106.0	Drama,Music	
tt5074352	Dangal	Dangal	2016	161.0	Action,Biography,Drama	
tt4633694	Spider-Man: Into the Spider-Verse	Spider-Man: Into the Spider-Verse	2018	117.0	Action,Adventure,Animation	
tt4154756	Avengers: Infinity War	Avengers: Infinity War	2018	149.0	Action,Adventure,Sci-Fi	
tt1345836	The Dark Knight Rises	The Dark Knight Rises	2012	164.0	Action,Thriller	
tt5311514	Your Name.	Kimi no na wa.	2016	106.0	Animation,Drama,Fantasy	
tt2380307	Coco	Coco	2017	105.0	Adventure,Animation,Comedy	
tt1853728	Django Unchained	Django Unchained	2012	165.0	Drama,Western	
tt1255953	Incendies	Incendies	2010	131.0	Drama,Mystery,War	
tt1832382	A Separation	Jodaeiye Nader az Simin	2011	123.0	Drama,Thriller	
tt0435761	Toy Story 3	Toy Story 3	2010	103.0	Adventure,Animation,Comedy	
tt1645089	Inside Job	Inside Job	2010	109.0	Crime,Documentary	
tt4430212	Drishyam	Drishyam	2015	163.0	Crime,Drama,Mystery	
tt4849438	Baahubali 2: The Conclusion	Baahubali 2: The Conclusion	2017	167.0	Action,Drama	

In [83]:

```
# Make the genres column values each a list, instead of one long string
highratings['genres'] = highratings['genres'].str.split(",")
```

In [84]:

```
# Check our work
highratings.head()
```

Out[84]:

tconst	primary_title	original_title	start_year	runtime_minutes	genres	averagerating	r
tt5813916	The Mountain II	Dag II	2016	135.0	[Action, Drama, War]	9.3	
tt1375666	Inception	Inception	2010	148.0	[Action, Adventure, Sci-Fi]	8.8	
tt4154796	Avengers: Endgame	Avengers: Endgame	2019	181.0	[Action, Adventure, Sci-Fi]	8.8	
tt0816692	Interstellar	Interstellar	2014	169.0	[Adventure, Drama, Sci-Fi]	8.6	
tt1424432	Senna	Senna	2010	106.0	[Biography, Documentary, Sport]	8.6	

In [85]:

```
# Flatten the DF
SingleGenre = (highratings
               .set_index(['primary_title','averagerating'])['genres']
               .apply(pd.Series)
               .stack()
               .reset_index()
               .drop('level_2', axis = 1)
               .rename(columns = {0 : 'genres'}))
```

In [86]:

```
SingleGenre.head()
```

Out[86]:

	primary_title	averagerating	genres
0	The Mountain II	9.3	Action
1	The Mountain II	9.3	Drama
2	The Mountain II	9.3	War
3	Inception	8.8	Action
4	Inception	8.8	Adventure

In [88]:

```
# Double check for placeholder values
# Should be rated from 1-10
SingleGenre['averagerating'].unique()
```

Out[88]:

```
array([9.3, 8.8, 8.6, 8.5, 8.4, 8.3, 8.2, 8.1, 8. , 7.9, 7.8, 7.7, 7.
6,
       7.5, 7.4, 7.3, 7.2, 7.1, 7. , 6.9, 6.8, 6.7, 6.6, 6.5, 6.4, 6.
3,
       6.2, 6.1, 6. , 5.9, 5.8, 5.7, 5.6, 5.5, 5.4, 5.3, 5.2, 5.1, 5.
,
       4.9, 4.8, 4.7, 4.6, 4.5, 4.4, 4.3, 4.2, 4.1, 3.5, 3.3, 2.2, 1.
8,
       1.6])
```

Plot & analyze

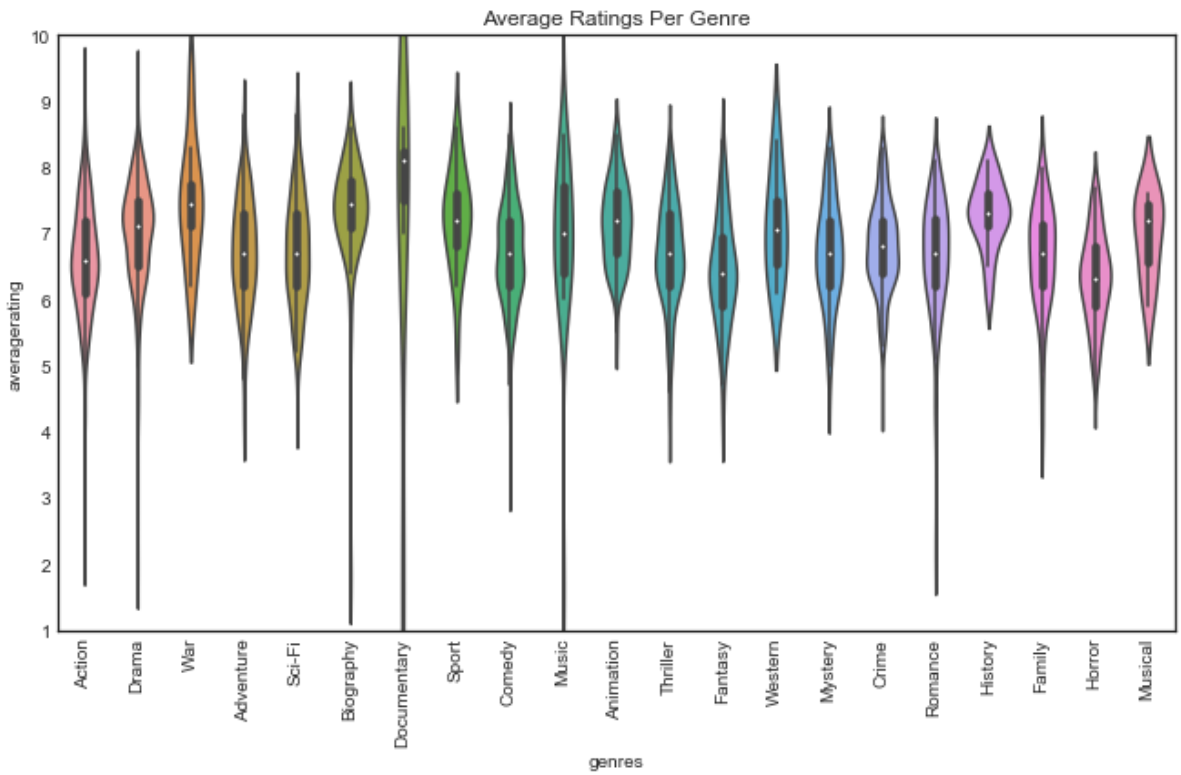
In [89]:

```
# Make a violin plot to see distribution of each genre's averageratings
f, ax = plt.subplots(figsize = (11, 6))
RatingVariation = sns.violinplot(data = SingleGenre,
              x = "genres",
              y = "averagerating")

RatingVariation.set_xticklabels(labels = RatingVariation.get_xticklabels(),
                                rotation = 90)
RatingVariation.set(ylim = (1,10))
RatingVariation.set_title('Average Ratings Per Genre')
```

Out[89]:

```
Text(0.5, 1.0, 'Average Ratings Per Genre')
```



In [602]:

```
# Calculate the averagerating's mean and median per genre
RatingsPerGenre = SingleGenre.groupby('genres')['averagerating'].agg(['mean'
```

In [603]:

```
RatingsPerGenre
```

Out[603]:

genres	mean	median
Documentary	7.114286	8.10
Biography	7.345000	7.45
War	7.470000	7.45
History	7.286667	7.30
Musical	6.957143	7.20
Animation	7.153521	7.20
Sport	7.205000	7.20
Drama	7.001553	7.10
Western	7.100000	7.05
Music	6.854545	7.00
Crime	6.800529	6.80
Adventure	6.719108	6.70
Family	6.643590	6.70
Mystery	6.719444	6.70
Romance	6.663235	6.70
Sci-Fi	6.731298	6.70
Thriller	6.681340	6.70
Comedy	6.690173	6.70
Action	6.631646	6.60
Fantasy	6.428829	6.40
Horror	6.322807	6.30

Question 3 Conclusion

To discover which types of films receive the highest ratings, we can look at the average ratings per ger
question was answered using the imdb.title.basics and imdb.title.ratings databases.

Upon joining these two tables, we can see a large variety of films ranging from 2010 to 2019, which is t
range we have looked at for the last 2 questions. I decided to limit the data set to include only the ratin
movies with 50,000 votes or more, since the average rating is highly skewed for movies with less votes
movie with 5 votes of all 9.0 ratings and up, will give a very high average rating based off little data).

Once we separate the movie genres into single-genre categories, we can look at the medians and distr of average ratings for each movie genre. This information tells us which movie genres generally receive ratings, and when looking at the violin plot, "Average Ratings Per Genre", we can also see the distribut each genre's average rating.

The movie genre with the highest median for "average rating" (the column name is average rating) is "Documentary", with a median rating of 8.10. Second highest is "Biography", with a median "average r 7.45. And tied for second highest is "War", with a median "average rating" of 7.45. However, there is nc variation between each median "average rating", as all of the medians fall between 6.30 and 8.10. Ther even though "Documentary", "Biography" and "War" movies are the highest rated movies, their ratings much higher than "Animation", for example, with a median "average rating" of 7.20.

What the violin plot, "Average Ratings Per Genre", does tell us, is the following:

- the long tails on violins indicate outliers in their ratings (e.g. Documentary, Biography, Music)
- the taller violins indicate genres with more varied ratings (e.g. War, Western, Sci-Fi)
- the wider violins indicate a higher probability that the rating will be in that range (e.g. History, Anim

Recommendations

Based on these findings, I would recommend making films with less varied ratings if we are looking to crowd-pleasing films.

Documentaries seem to be extremely varied, so we should avoid making those films, whereas Animate seem to be well-liked with less variation in ratings.

Question 4: When should we release our films?

Which release months make the most money?

Prepare the DF we need

Movie_releases was already cleaned, but we can drop the columns we don't need to make it easier to

In [539]:

```
# Get rid of the columns we won't use to answer this question
movie_releases = movie_budgets.drop('production_budget', axis = 1)
```

In [541]:

```
movie_releases = movie_releases.drop('worldwide_gross', axis = 1)
```

In [543]:

```
movie_releases = movie_releases.drop('roi', axis = 1)
```

In [551]:

```
# Quick preview of release dates with highest domestic gross
movie_releases.groupby('release_date')['domestic_gross'].sum().sort_values(a
```

Out[551]:

```
release_date
2015-12-18    1111370900
2013-11-22     893775852
2018-02-16     708327110
2017-12-15     704591762
2018-04-27     678815482
                  ...
2015-03-17             0
2011-06-28             0
2015-03-24             0
2011-06-21             0
2019-11-22             0
Name: domestic_gross, Length: 726, dtype: int64
```

In [562]:

```
# Create column indicating day of the week of release
movie_releases['day'] = movie_releases['release_date'].dt.day_name()
```

In [563]:

```
# Create column indicating month of release
movie_releases['month'] = movie_releases['release_date'].dt.month
```

In [587]:

```
movie_releases.sort_values(by = 'domestic_gross', ascending = False).head(20
```

Out[587]:

	id	release_date	movie	domestic_gross	day	month	holiday
5	6	2015-12-18	Star Wars Ep. VII: The Force Awakens	936662225	Friday	12	False
41	42	2018-02-16	Black Panther	700059566	Friday	2	False
6	7	2018-04-27	Avengers: Infinity War	678815482	Friday	4	False
33	34	2015-06-12	Jurassic World	652270625	Friday	6	False
26	27	2012-05-04	The Avengers	623279547	Friday	5	False
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	620181382	Friday	12	False
43	44	2018-06-15	Incredibles 2	608581744	Friday	6	False
44	45	2016-12-16	Rogue One: A Star Wars Story	532177324	Friday	12	False
134	35	2017-03-17	Beauty and the Beast	504014165	Friday	3	False
45	46	2016-06-17	Finding Dory	486295561	Friday	6	False
3	4	2015-05-01	Avengers: Age of Ultron	459005868	Friday	5	False
10	11	2012-07-20	The Dark Knight Rises	448139099	Friday	7	False
95	96	2019-03-08	Captain Marvel	426525952	Friday	3	False
237	38	2013-11-22	The Hunger Games: Catching Fire	424668047	Friday	11	False
112	13	2018-06-22	Jurassic World: Fallen Kingdom	417719760	Friday	6	False
46	47	2010-06-18	Toy Story 3	415004880	Friday	6	False
154	55	2017-06-02	Wonder Woman	412563408	Friday	6	False
47	48	2013-05-03	Iron Man 3	408992272	Friday	5	False
16	17	2016-05-06	Captain America: Civil War	408084349	Friday	5	False
537	38	2012-03-23	The Hunger Games	408010692	Friday	3	False

In [558]:

```
# Let's import this calendar function to find the releases on holidays
from pandas.tseries.holiday import USFederalHolidayCalendar
```

In [561]:

```
cal = USFederalHolidayCalendar()

# Use cal.holidays to set our start and end date to calculate holidays and m
holidays = cal.holidays(start = movie_releases['release_date'].min(),
                        end = movie_releases['release_date'].max()).to_pydat

# Create column with booleans indicating holiday or no holiday
movie_releases['holiday'] = movie_releases['release_date'].isin(holidays)
```

Sort, plot, and analyze

In [568]:

```
# Sort values by highest domestic gross
mr = movie_releases.sort_values('domestic_gross', ascending = False)
mr.shape
```

Out[568]:

```
(2185, 7)
```

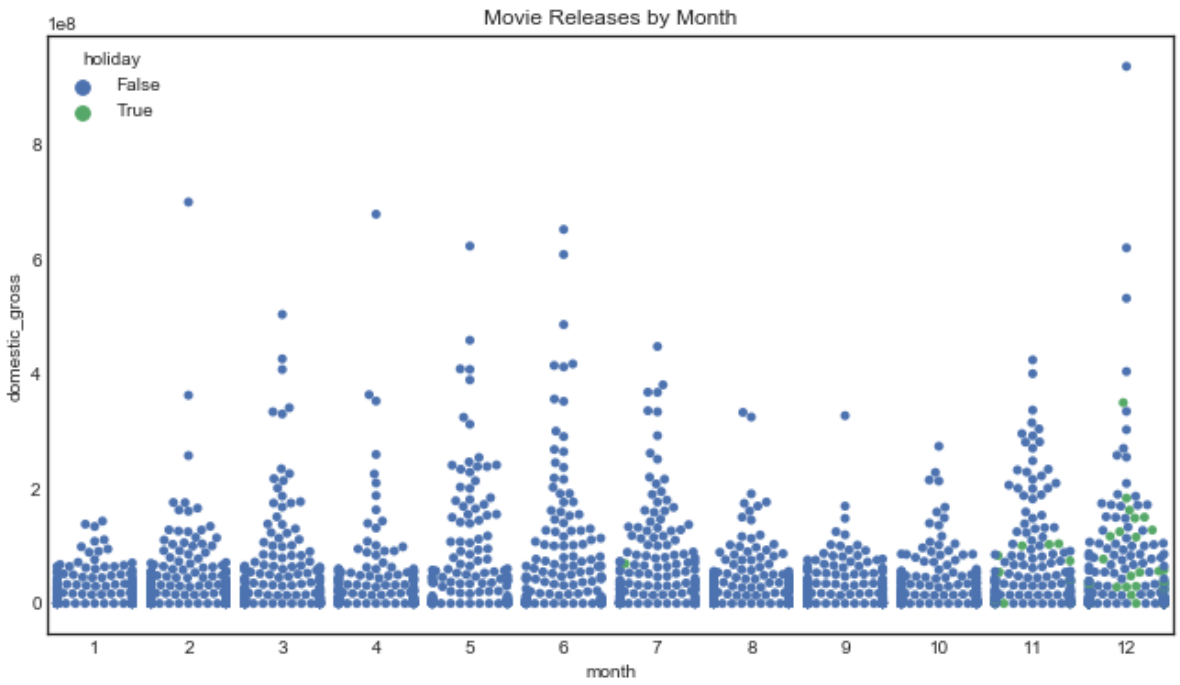
In [653]:

```
# Plot to see the movies that were released at different months
# and if they were released on holidays

f, ax = plt.subplots(figsize = (11, 6))
m3 = sns.swarmplot(data = mr,
                   x = "month",
                   y = "domestic_gross",
                   hue = "holiday")
m3.set_title('Movie Releases by Month')
```

Out[653]:

```
Text(0.5, 1.0, 'Movie Releases by Month')
```



In [594]:

```
# Group movie releases by month and find mean and median domestic gross for
mr_by_month = movie_releases.groupby('month')['domestic_gross'].agg(['mean',
```

In [597]:

```
# Sort by median, since that number is less affected by outliers
mr_by_month.sort_values(by = 'median', ascending = False)
```

Out[597]:

month	mean	median
7	6.357822e+07	31206263.0
11	6.460815e+07	30659817.0
8	3.471135e+07	21295021.0
5	6.749728e+07	20316694.0
2	4.177002e+07	19452138.0
1	2.647812e+07	18504178.5
6	6.982529e+07	16847261.0
3	4.536472e+07	12490404.5
9	2.479126e+07	8005586.0
10	2.666106e+07	6393616.5
4	3.249267e+07	4352828.5
12	4.261963e+07	1434498.0

In [598]:

```
# Compare to when we sort by mean
mr_by_month.sort_values(by = 'mean', ascending = False)
```

Out[598]:

month	mean	median
6	6.982529e+07	16847261.0
5	6.749728e+07	20316694.0
11	6.460815e+07	30659817.0
7	6.357822e+07	31206263.0
3	4.536472e+07	12490404.5
12	4.261963e+07	1434498.0
2	4.177002e+07	19452138.0
8	3.471135e+07	21295021.0
4	3.249267e+07	4352828.5
10	2.666106e+07	6393616.5
1	2.647812e+07	18504178.5
9	2.479126e+07	8005586.0

Question 4 Conclusion

To answer this question, the tn.movie_budgets database was helpful in providing movie release dates, as gross domestic earnings. We used domestic gross over worldwide gross since holidays vary from c country. With this database, I included movies from 2010-2020, which gives us data from 10 occurrenc each month.

When we look at the domestic gross by month over the past 10 years, we can find which months have highest domestic gross. We can use the median domestic gross per month as a good indicator to rule outliers. Thus, the highest grossing months are:

- July - $31,206,263
- November - $30,659,817
- August - $21,295,021
- April - $20,316,694
- February - $19,452,138

Whether or not you release the movie directly on a holiday (i.e. Thanksgiving Day) does not really affect gross. In the plot, "Movie Releases by Month", we can see lots of green dots in December, indicating t movie was released on a holiday, yet December had the lowest median domestic gross earnings.

Recommendations

Based on these findings, I would recommend releasing our films over the summer, or November. Febru seems to be a great time for releases.

I would also recommend NOT releasing a film in December, because it had the lowest domestic gross month. This could be due to the fact that many other studios release films during this time, or because are generally busy with family, travel, etc over this time, but it would take more research and data to understand why this happens.

In [ ]: