Final Phase One Project Submission

- Student name: Rose Wairimu Kimondo
- Student pace: Part Time

Overview

The task at hand is to assist Microsoft in their new venture of creating a movie studio and identify the types of films that are doing well at the Box Office.

My analysis of the movie industry which was done by gathering data from IMDB, The Numbers, and The Box Office Mojo and producing descriptive statistics and visualizations finds that:

- **Genre:** The highest grossing films made by the leading studios are animated, superhero, scifi, and fantasy movies, which should be the genres that the new studio should focus on.
- **Production Budget:** There is a very strong relationship between the production budget and ROI. If the studio has a lot of money to invest, they should choose a sci-fi/fantasy/superhero franchise film. However, with a small budget, they can choose a horror film and still get a high ROI.
- **Release Month:** Based on the analysis, the four months with the highest earnings based on domestic and worldwide returns are July, November, August, and February. Microsoft should focus on releasing movies during these months.

Microsoft can use this report to target their production budget, genre, and release time to generate the highest amount of revenue possible.

Business Understanding

> Microsoft sees the allure of the film business and feels that they also need to get into the market. The decision involves the creation of a new movie studio.
>
> However, there is a limitation surrounding information with little awareness of what to do, how to enter the market, what movies to make, when to make them, which studios to emulate, how much money to invest, and a lot of other crucial matters that need understanding before any other steps can be taken.
>
> > The task at hand is to explore the types of films that are doing well at tbhe Box Office and translate these findings into useful insights to help Microsoft decide on the way forward.

The topics informing the questions for this analysis are:

- Genre
- Release month
- Production budget

Data Questions:

1. a. Which studios make the highest grossing films?
   b. What are the genres of the highest grossing films that are made by these studios?

2. a. How does spending on production translate to Gross Earnings? Does higher spending lead to higher earnings?

   b. What is the relationship between the production budget and the gross earnings?
   c. Which genre has the highest return on Investment?

3. When should the films be released? Which release months make the most money?

4. Which movie genres receive the highest ratings?

Data Understanding

I used three different data sources for my analysis to have a comprehensive view of the industry:

- The Numbers Movie Budget- The most important information from this dataset were the release dates, production budget, domestic gross, and worldwide gross earnings. These were used to determine the relationship between the production budget and the earnings, and the release month and the earnings.

- The Box Office Mojo movie gross earnings: This data was used to determine the studios that have the highest grossing movies and then identify the genres that the top grossing movies from the studios belong to.

- IMDB- The data was used to analyze the ratings for different genres and affirm our conclusions from the Box Office Mojo data.

Data Importation & Preparation

In [115…

```
# Import standard packages
import pandas as pd
import numpy as np
import os
import sqlite3
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

In [115…

```
# Open up a connection
conn = sqlite3.connect('data/im.db')
```

In [115…

```
# Viewing the list of tables in the IMDB Database
table_name_query = """SELECT *
                      FROM sqlite_master;
"""

pd.read_sql(table_name_query, conn).head()
```

Out[1152]:

	type	name	tbl_name	rootpage	sql
0	table	movie_basics	movie_basics	2	CREATE TABLE "movie_basics" (\n"movie_id" TEXT...
1	table	directors	directors	3	CREATE TABLE "directors" (\n"movie_id" TEXT,\n...
2	table	known_for	known_for	4	CREATE TABLE "known_for" (\n"person_id" TEXT,\...
3	table	movie_akas	movie_akas	5	CREATE TABLE "movie_akas" (\n"movie_id" TEXT,\...
4	table	movie_ratings	movie_ratings	6	CREATE TABLE "movie_ratings" (\n"movie_id" TEX...

In [115…

```
# Viewing the tables in the df
imdb_tables = pd.read_sql("""SELECT name FROM sqlite_master WHERE type = 'table';""", co
imdb_tables
```

Out[1153]:

	name
0	movie_basics
1	directors
2	known_for
3	movie_akas
4	movie_ratings
5	persons
6	principals
7	writers

In [115…

```
#Viewing the columns in the movie basics table
movie_basics = """SELECT * FROM movie_basics;"""

pd.read_sql(movie_basics, conn)
```

Out[1154]:

	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 [115…

```
# Read the Box Office Mojo Table Data
# The table seems to contain useful information for domestic and gross earnings and the
bom_movie_info = pd.read_csv('data/bom.movie_gross.csv')
bom_movie_info.head()
```

Out[1155]:

	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 [115…

```
bom_movie_info.shape
```

Out[1156]:

```
(3387, 5)
```

In [115…

```
# Finding the number of unique studios. There are 258
len(bom_movie_info['studio'].unique())
```

Out[1157]:

```
258
```

In [115…

```
# Exploring the year ranges for the data.
# It starts from 2020 to 2018
bom_movie_info['year'].unique()
```

Out[1158]:

```
array([2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018], dtype=int64)
```

In [115…

```
# Read The Numbers Table Data, which provides perfect data for comparing
# production budget and gross earnings
# We will call it the budget&earnings table
budget_and_earnings = pd.read_csv('data/tn.movie_budgets.csv')
budget_and_earnings.head()
```

Out[1159]:

	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 [116…

```
# Exploring the budget and earnings df
budget_and_earnings['release_date'].unique
```

Out[1160]:

```
<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 [116…

```
# Exploring the budget and earnings df
budget_and_earnings.shape
```

Out[1161]:

```
(5782, 6)
```

Data Analysis

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

Which studios are producing the highest domestic grossing films?

Clean/prepare the table (We will use the BOM table(bom_movie_info) since it contains details for studios which can be compared with the gross earnings)

- Get rid of null values
- Get rid of duplicates

Because we are just starting the movie production business, we will focus on domestic gross

In [116…

```
# Check for null values in the domestic_gross column
null_domestic_gross_values = bom_movie_info['domestic_gross'].isna().sum()/len(bom_movie
print("The percentage of null domestic gross values is:", null_domestic_gross_values, "%
```

```
The percentage of null domestic gross values is: 0.8266902863891349 % which is insignifi
cant.
```

In [116…

```
# This is only 0.8% of the values, which is a low number.
# Let us take a look at the data and see whether we will be dropping titles that
# will affect the domestic gross function
bom_movie_info.loc[bom_movie_info['domestic_gross'].isna() == True]
# From the results, most of these are foreign titles and they are not too many, and we c
```

Out[1163]:

	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
936	Lula, Son of Brazil	NYer	NaN	3800000	2012

966	The Cup (2012)	Myr.	NaN	1800000	2012
1017	Dark Tide	WHE	NaN	432000	2012
1079	The Green Wave	RF	NaN	70100	2012
1268	22 Bullets	Cdgm.	NaN	21300000	2013
1308	Matru Ki Bijlee Ka Mandola	FIP	NaN	6000000	2013
1340	The Snitch Cartel	PI	NaN	2100000	2013
1342	All the Boys Love Mandy Lane	RTWC	NaN	1900000	2013
1368	6 Souls	RTWC	NaN	852000	2013
1659	Jessabelle	LGF	NaN	7000000	2014
1681	14 Blades	RTWC	NaN	3800000	2014
1685	Jack and the Cuckoo-Clock Heart	Shout!	NaN	3400000	2014
1739	Lila Lila	Crnth	NaN	1100000	2014
1975	Surprise - Journey To The West	AR	NaN	49600000	2015
2392	Finding Mr. Right 2	CL	NaN	114700000	2016
2468	Solace	LGP	NaN	22400000	2016
2595	Viral	W/Dim.	NaN	552000	2016
2825	Secret Superstar	NaN	NaN	122000000	2017

In [116…

```
# Check for null values in the studio column
# We can drop all of these since thy are only five, which is equal to
bom_movie_info.loc[bom_movie_info['studio'].isna() == True]
null_studio_values = bom_movie_info['studio'].isna().sum()/len(bom_movie_info['studio'])
print("The percentage of null studio values is:", null_studio_values, "% which is insign
```

```
The percentage of null studio values is: 0.14762326542663123 % which is insignificant.
```

In [116…

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

In [116…

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

Out[1166]:

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

In [116…

```
# Look at nulls in the foreign gross column since there are a lot of them
bom_gross.loc[bom_gross['foreign_gross'].isna() == True]
null_foreign_gross = bom_movie_info['foreign_gross'].isna().sum()/len(bom_movie_info['fo
print("The percentage of null foreign gross values is:", null_foreign_gross, "% which is

# 40% is very large, and further affirms our decision to only use the domestic gross val
# We can drop the foreign gross column
```

```
The percentage of null foreign gross values is: 39.85828166519043 % which is insignifica
nt.
```

In [116…

```
bom_gross = bom_gross.drop(columns = 'foreign_gross')
```

In [116…

```
# Check again for nulls. There are now no nulls
bom_gross.isna().sum()
```

Out[1169]:
```
title             0
studio            0
domestic_gross    0
year              0
dtype: int64
```

In [117…
```
# Checking for duplicates
duplicates = bom_gross[bom_gross.duplicated()]
print(len(duplicates))

# There are no duplicates
```

```
0
```

In [117…
```
# Checking 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 off
bom_gross['domestic_gross'].value_counts()
```

Out[1171]:
```
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 [117…
```
# Making sure years are within our desired range
# This is still 2010 to 2018
bom_gross['year'].value_counts()
```

Out[1172]:
```
2015    449
2016    433
2011    396
2012    393
2014    390
2013    345
2010    322
2017    320
2018    308
Name: year, dtype: int64
```

Visualizing

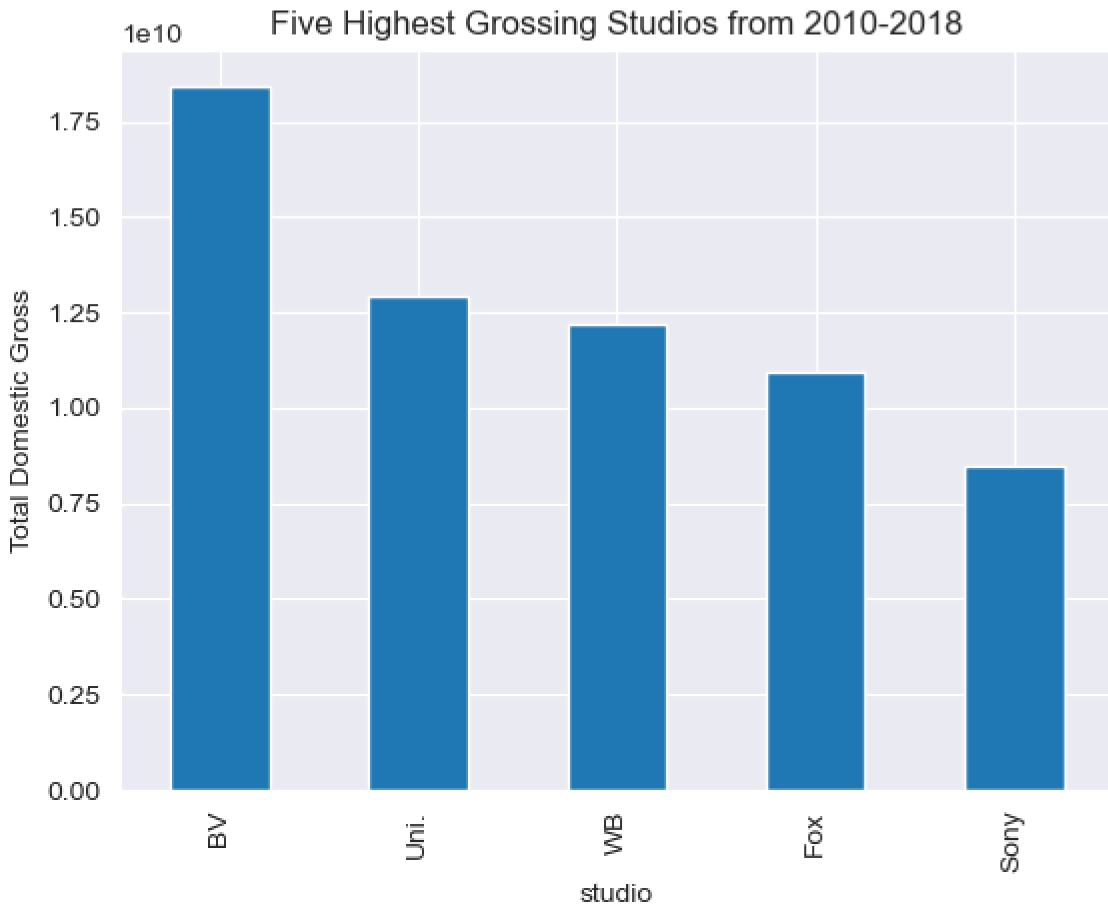
In [117…
```
sns.set_style('darkgrid')
```

In [117…
```
# Group the data by studio and display the total that each studio made
bom_studios = bom_gross.groupby('studio')['domestic_gross'].sum().sort_values(ascending
bom_studios.head()
```

Out[1174]:
```
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 [117…
```
# Create the plot showing the Five highest grossing studios, representing the best perfo
studios_plt = bom_studios.head().plot(kind = 'bar')
```

```
plt.title('Five Highest Grossing Studios from 2010-2018')
plt.ylabel('Total Domestic Gross');
```



Further Evaluation: What are the genres of these highest grossing films that the studios are making?

We can sort the films based on domestic gross and determine the top 10 films per studio for the five leading studios

Please remember that these are films from 2010 to 2018

In [117…

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

1. Buena Vista (BV)

In [117…

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

Out[1177]:

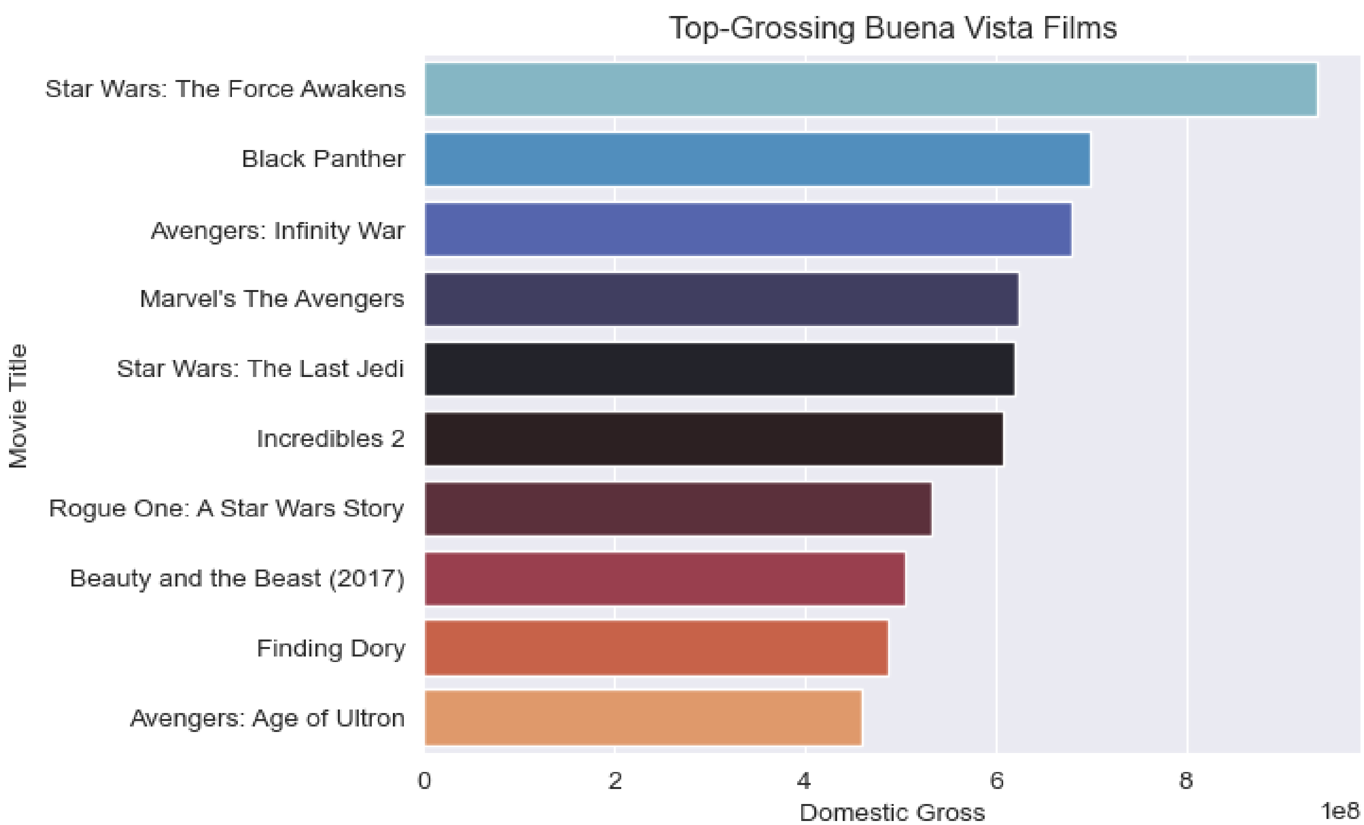
	title	studio	domestic_gross	year
1872	Star Wars: The Force Awakens	BV	936700000.0	2015
3080	Black Panther	BV	700100000.0	2018
3079	Avengers: Infinity War	BV	678800000.0	2018
727	Marvel's The Avengers	BV	623400000.0	2012
2758	Star Wars: The Last Jedi	BV	620200000.0	2017
3082	Incredibles 2	BV	608600000.0	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

Visualizing

In [117…

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

BV_plot.set_title('Top-Grossing Buena Vista Films')
BV_plot.set_ylabel('Movie Title')
BV_plot.set_xlabel('Domestic Gross')
BV_plot;
```



2. Universal

In [117…

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

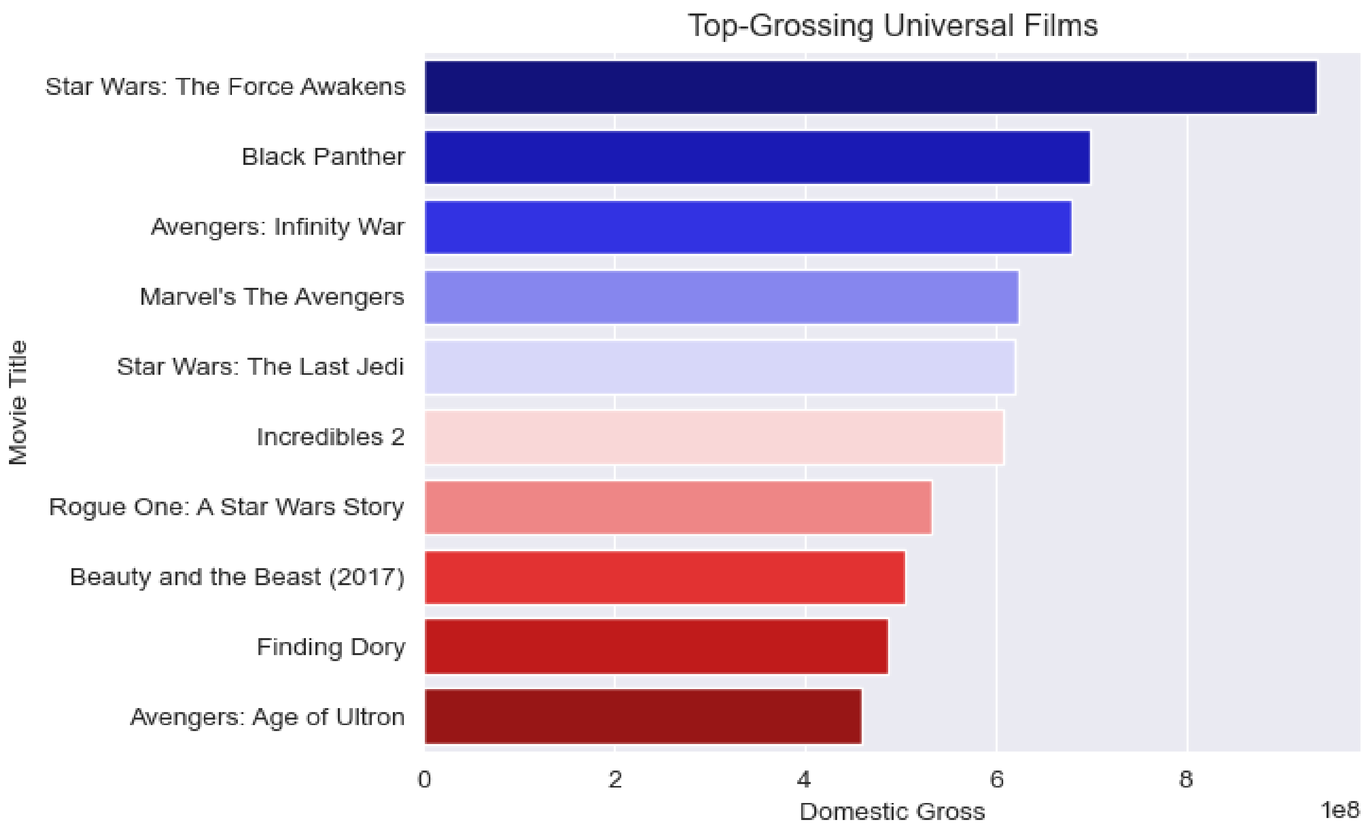
Out[1179]:

	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 [118…

```
# Plot the top grossing films for Universal
Uni_plot = sns.barplot(data = top_BV.head(10),
                      x = 'domestic_gross',
                      y = 'title',
                     palette = 'seismic')

Uni_plot.set_title('Top-Grossing Universal Films')
Uni_plot.set_ylabel('Movie Title')
Uni_plot.set_xlabel('Domestic Gross')
Uni_plot;
```



3. Warner Bros

In [118…

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

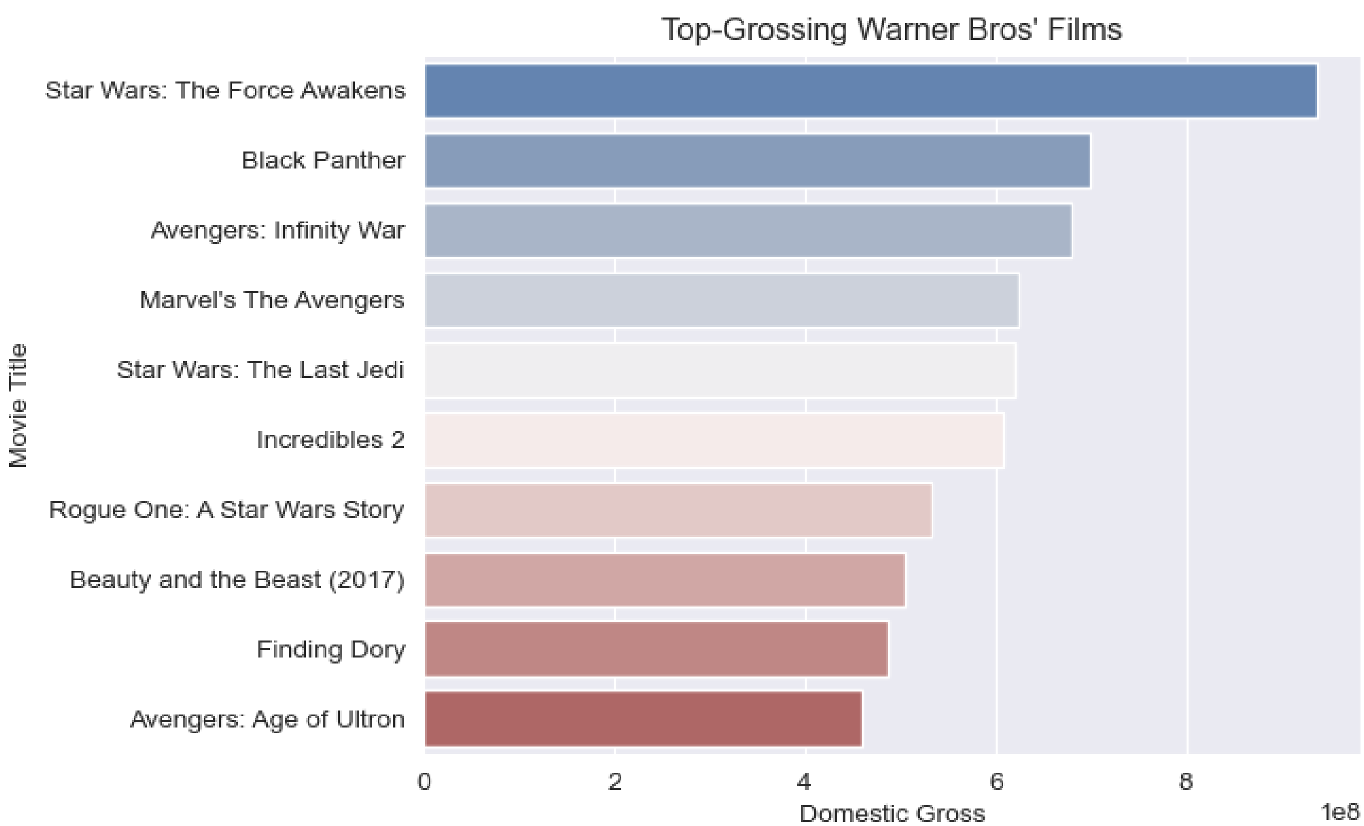
Out[1181]:

	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 [118…

```
# Plot the top grossing films for Warner Bros
WB_plot = sns.barplot(data = top_BV.head(10),
                      x = 'domestic_gross',
                      y = 'title',
                     palette = 'vlag')

WB_plot.set_title("Top-Grossing Warner Bros' Films")
WB_plot.set_ylabel('Movie Title')
WB_plot.set_xlabel('Domestic Gross')
WB_plot;
```



4. Fox

In [118…

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

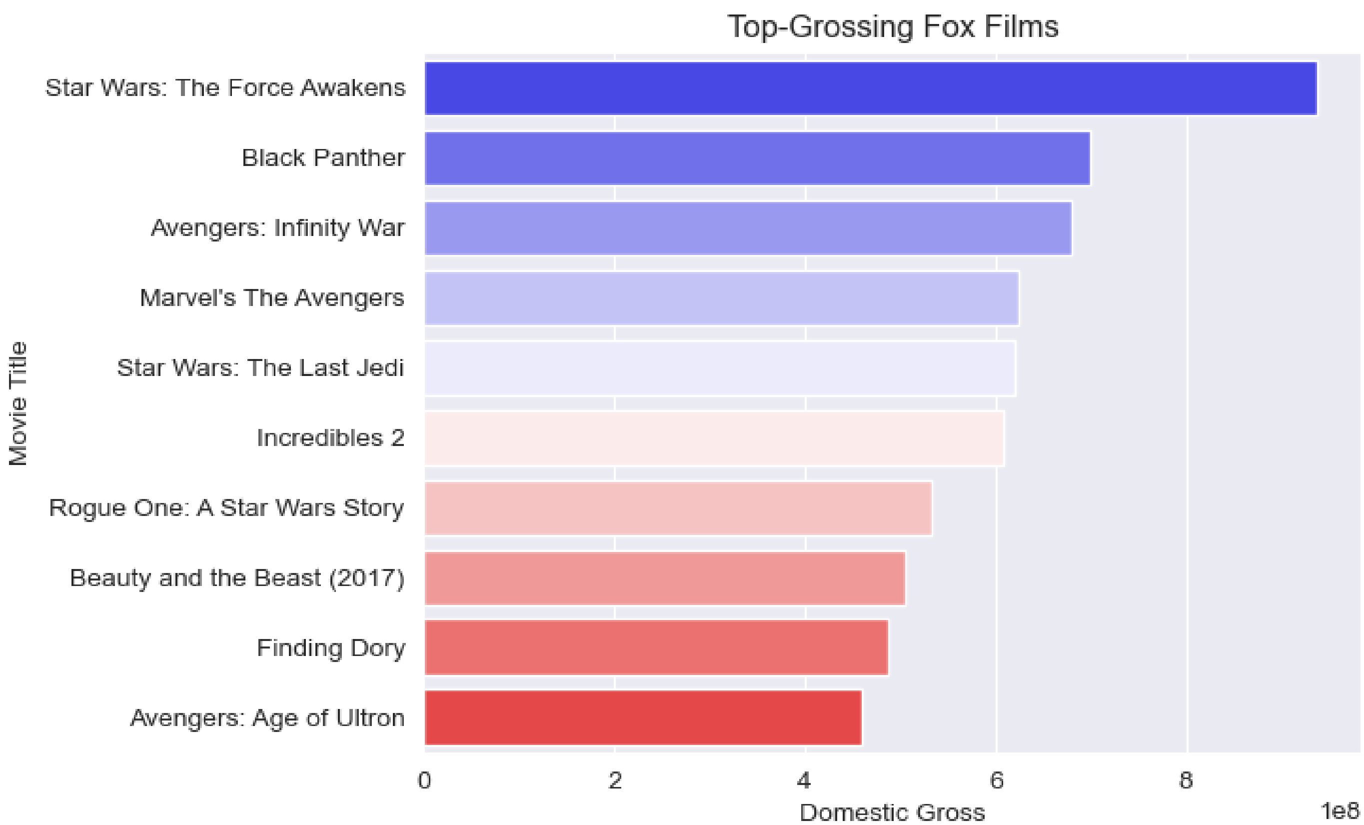
Out[1183]:

	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

	title	studio	domestic_gross	year
1137	The Croods	Fox	187200000.0	2013
1884	The Revenant	Fox	183600000.0	2015
1890	Home (2015)	Fox	177400000.0	2015

In [118…

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

Fox_plot.set_title('Top-Grossing Fox Films')
Fox_plot.set_ylabel('Movie Title')
Fox_plot.set_xlabel('Domestic Gross')
Fox_plot;
```



5. Sony

In [118…

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

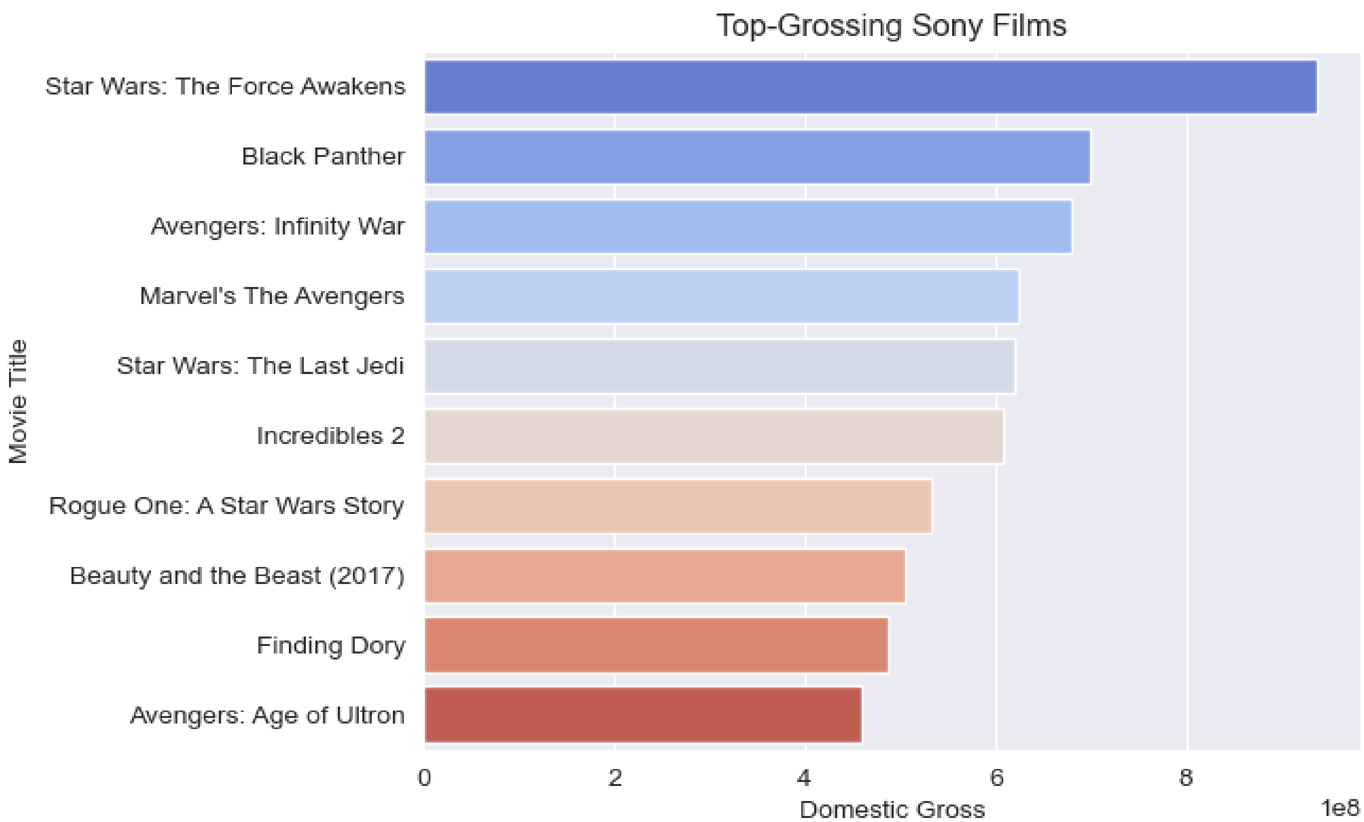
Out[1185]:

	title	studio	domestic_gross	year
2762	Jumanji: Welcome to the Jungle	Sony	404500000.0	2017
2763	Spider-Man: Homecoming	Sony	334200000.0	2017
728	Skyfall	Sony	304400000.0	2012
733	The Amazing Spider-Man	Sony	262000000.0	2012
3085	Venom (2018)	Sony	213500000.0	2018
1485	The Amazing Spider-Man 2	Sony	202900000.0	2014
1877	Spectre	Sony	200100000.0	2015
1502	22 Jump Street	Sony	191700000.0	2014

3102	Spider-Man: Into The Spider-Verse	Sony	190200000.0	2018
736	MIB 3	Sony	179000000.0	2012

In [118…

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

Sony_plot.set_title('Top-Grossing Sony Films')
Sony_plot.set_ylabel('Movie Title')
Sony_plot.set_xlabel('Domestic Gross')
Sony_plot;
```



Question 1 Conclusion

What are the genres of the top-grossing films from 2010 to 2018?

To conclude, we looked at the bom_movie_gross dataset from Box Office Mojo.

The top-grossing films were defined as those with the highest gross earnings.

Domestic gross was chosen over foreign gross because as a movie studio, it is important to conquer the home market before venturing into foreign markets.

The top five 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
- Sony | total domestic gross: $8.4 billion

We then further investigated the top grossing studios by looking at the top 10 films for each of the studios.

The plots referenced above are `BV_plot, Uni_plot, WB_plot, Fox_plot, and Sony_plot`

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

RECOMMENDATIONS

Based on these findings, the recommendation is to make films that have these qualities (animated, superhero, scifi, fantasy).

Investing in the scifi/fantasy and superhero franchises seems to be a good idea as we can see a positive trend for these films from 2010 - 2018.

It can also be concluded that benchmarking these five studios (Buena Vista, Universal, Warner Bros, Fox, and Sony) will be an excellent idea for identifying their best practices to emulate them

Question 2. How does spending on production translate to Gross Earnings? Does higher spending lead to higher earnings?

What is the relationship between the production budget and the gross earnings?

Clean and prepare the table (We will use The Numbers' Movie Budgets Data)

- Check for null values
- Check for duplicates
- Change the datatypes as required
- Limit the table to a reasonable year range

In [118…

```
# Looking for duplicates in the table
# Check for duplicates
# There are no dublicates
mb_duplicates = budget_and_earnings[budget_and_earnings.duplicated()]
print(len(mb_duplicates))
```

```
0
```

In [118…
```
# Looking for null values. There are no null values
budget_and_earnings.isna().sum()
```

Out[1188]:
```
id                   0
release_date         0
movie                0
production_budget    0
domestic_gross       0
worldwide_gross      0
dtype: int64
```

In [118…
```
# Confirming the data types of the production and gross columns since we want numbers
budget_and_earnings.dtypes
# These are currently not integers and will have to be converted
```

Out[1189]:
```
id                    int64
release_date         object
movie                object
production_budget    object
domestic_gross       object
worldwide_gross      object
dtype: object
```

In [119…
```
# It has been noted that the release date is an object and we will use it later
# Therefore, convert it to a datetime object
budget_and_earnings['release_date'] = pd.to_datetime(budget_and_earnings['release_date']
budget_and_earnings.head()
```

Out[1190]:

	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 [119…
```
# Change production budget to an integrer
budget_and_earnings['production_budget'] = budget_and_earnings['production_budget'].str.
```

In [119…
```
# Change domestic gross to an integer
budget_and_earnings['domestic_gross'] = budget_and_earnings['domestic_gross'].str.replac
```

In [119…
```
# Change Worldwide Gross to an integer
budget_and_earnings['worldwide_gross'] = budget_and_earnings['worldwide_gross'].str.repl
budget_and_earnings.head()
```

Out[1193]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	2009-12-18	Avatar	425000000	760507625	2.776345e+09
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1.045664e+09
2	3	2019-06-07	Dark Phoenix	350000000	42762350	1.497624e+08
3	4	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1.403014e+09
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1.316722e+09

In [119…
```
budget_and_earnings.dtypes
```

Out[1194]:
```
id                             int64
```

```
release_date          datetime64[ns]
movie                         object
production_budget              int32
domestic_gross                 int32
worldwide_gross              float64
dtype: object
```

In [119…

```
budget_and_earnings.head()
```

Out[1195]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	2009-12-18	Avatar	425000000	760507625	2.776345e+09
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1.045664e+09
2	3	2019-06-07	Dark Phoenix	350000000	42762350	1.497624e+08
3	4	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1.403014e+09
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1.316722e+09

In [119…

```
# Looking at the range of years in our data
print(budget_and_earnings['release_date'].dt.year.unique())
```

```
[2009 2011 2019 2015 2017 2018 2007 2012 2013 2010 2016 2014 2006 2008
 2005 1997 2004 1999 1995 2003 2001 2020 2002 1998 2000 1991 1994 1996
 1993 1992 1988 1990 1989 1978 1981 1984 1982 1985 1980 1963 1987 1986
 1983 1979 1977 1970 1969 1976 1965 1962 1964 1959 1966 1974 1956 1975
 1973 1960 1967 1968 1971 1951 1972 1961 1946 1944 1953 1954 1957 1952
 1930 1939 1925 1950 1948 1958 1943 1940 1945 1947 1938 1927 1949 1955
 1936 1937 1941 1942 1933 1935 1931 1916 1929 1934 1915 1920]
```

In [119…

```
# We can pick 2010 to 2019 (which is the latest year)
# This will make our analysis in line with other questions which are in a similar range
budget_and_earnings = budget_and_earnings.loc[(budget_and_earnings['release_date'] > '20
                                  (budget_and_earnings['release_date'] < '2019-12-31')]
```

In [119…

```
budget_and_earnings.head()
```

Out[1198]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1.045664e+09
2	3	2019-06-07	Dark Phoenix	350000000	42762350	1.497624e+08
3	4	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1.403014e+09
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1.316722e+09
5	6	2015-12-18	Star Wars Ep. VII: The Force Awakens	306000000	936662225	2.053311e+09

In [119…

```
budget_and_earnings.dtypes
```

Out[1199]:

```
id                             int64
release_date          datetime64[ns]
movie                         object
production_budget              int32
domestic_gross                 int32
worldwide_gross              float64
dtype: object
```

Visualizing

In [120…

```
# Sort by the smallest and largest production budgets
```

```
budget_and_earnings.sort_values(by = 'production_budget')
```

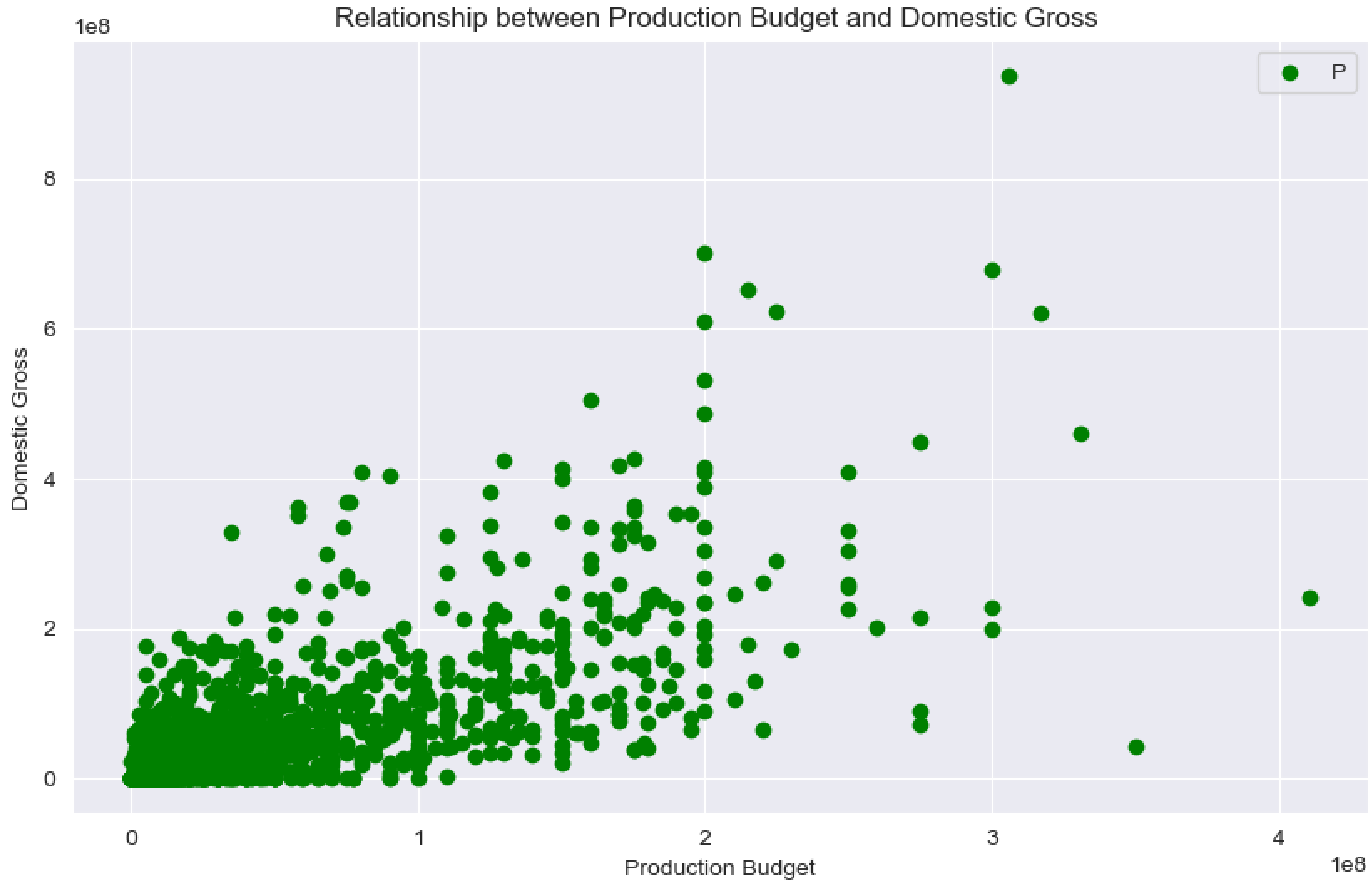
Out[1200]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
5780	81	2015-09-29	A Plague So Pleasant	1400	0	0.000000e+00
5777	78	2018-12-31	Red 11	7000	0	0.000000e+00
5772	73	2012-01-13	Newlyweds	9000	4584	4.584000e+03
5771	72	2015-05-19	Family Motocross	10000	0	0.000000e+00
5760	61	2010-04-02	Breaking Upwards	15000	115592	1.155920e+05
...	...	...	...	...	...	...
5	6	2015-12-18	Star Wars Ep. VII: The Force Awakens	306000000	936662225	2.053311e+09
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1.316722e+09
3	4	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1.403014e+09
2	3	2019-06-07	Dark Phoenix	350000000	42762350	1.497624e+08
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1.045664e+09

2185 rows × 6 columns

In [120…

```
# A scatter plot showing the relationship between production budget and domestic gross
scatter_budget_and_earnings, ax = plt.subplots(figsize=(10,6))

# Plot with scatter()
ax.scatter(x = budget_and_earnings['production_budget'], y = budget_and_earnings['domest

plt.title('Relationship between Production Budget and Domestic Gross ')
# Legend
plt.legend('Production Budget')
# Labels
plt.xlabel('Production Budget')
plt.ylabel('Domestic Gross');
```



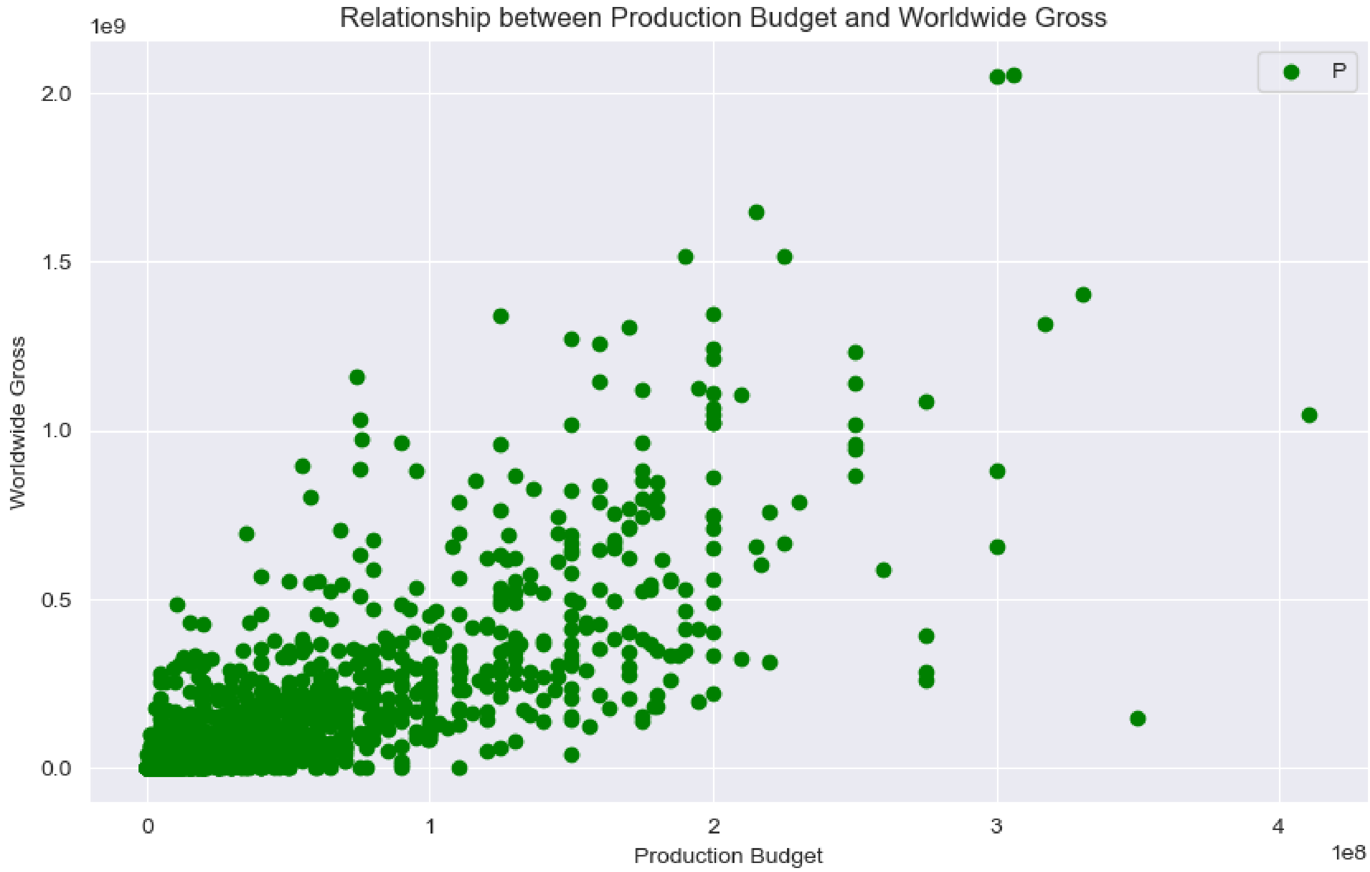
In [120…
```
# We can calculate the Correlation coefficient to make a solid conclusion
# From the scatter plot and the coefficient
np.corrcoef(x = budget_and_earnings['production_budget'],
            y = budget_and_earnings['domestic_gross'])
```

Out[1202]:
```
array([[1.        , 0.73482153],
       [0.73482153, 1.        ]])
```

In [120…
```
# Compare the relationship with Worldwide Gross
np.corrcoef(x = budget_and_earnings['production_budget'],
            y = budget_and_earnings['worldwide_gross'])
```

Out[1203]:
```
array([[1.        , 0.79646255],
       [0.79646255, 1.        ]])
```

In [120…
```
# Make the same plot, but for worldwide gross
scatter_budget_and_earnings, ax = plt.subplots(figsize=(10,6))

# Plot with scatter()
ax.scatter(x = budget_and_earnings['production_budget'], y = budget_and_earnings['worldw

plt.title('Relationship between Production Budget and Worldwide Gross ')
# Legend
plt.legend('Production Budget')
# Labels
plt.xlabel('Production Budget')
plt.ylabel('Worldwide Gross');
```



Further Evaluation: Which films have the highest Return on Investment

This refers to the profits which we will present in percentage form

In [120…

```
budget_and_earnings['film_roi'] = ((budget_and_earnings['worldwide_gross'] - budget_and_
budget_and_earnings
```

Out[1205]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	film_roi
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1.045664e+09	154.667286
2	3	2019-06-07	Dark Phoenix	350000000	42762350	1.497624e+08	-57.210757
3	4	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1.403014e+09	324.384139
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1.316722e+09	315.369636
5	6	2015-12-18	Star Wars Ep. VII: The Force Awakens	306000000	936662225	2.053311e+09	571.016739
...	...	...	...	...	...	...	...
5761	62	2014-12-31	Stories of Our Lives	15000	0	0.000000e+00	-100.000000
5771	72	2015-05-19	Family Motocross	10000	0	0.000000e+00	-100.000000
5772	73	2012-01-13	Newlyweds	9000	4584	4.584000e+03	-49.066667
5777	78	2018-12-31	Red 11	7000	0	0.000000e+00	-100.000000

5780	81	2015-09-29	A Plague So Pleasant	1400	0	0.000000e+00	-100.000000

2185 rows × 7 columns

In [120…

```
# Sort films by highest ROI
sorted_ROI = budget_and_earnings.sort_values(ascending = False, by = 'film_roi')
sorted_ROI.head(10)
```

Out[1206]:

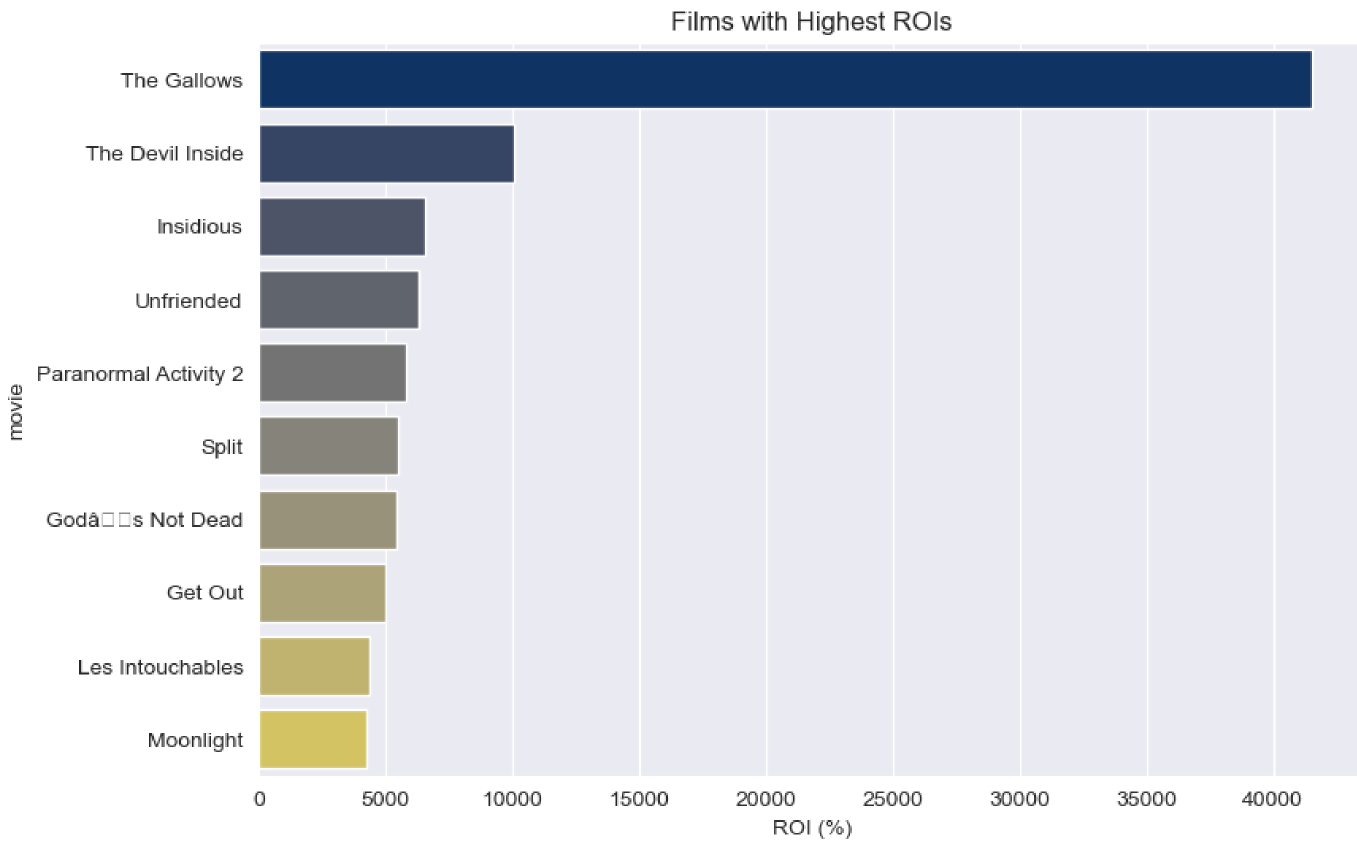
	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	film_roi
5679	80	2015-07-10	The Gallows	100000	22764410	41656474.0	41556.474000
5211	12	2012-01-06	The Devil Inside	1000000	53262945	101759490.0	10075.949000
5062	63	2011-04-01	Insidious	1500000	54009150	99870886.0	6558.059067
5213	14	2015-04-17	Unfriended	1000000	32789645	64364198.0	6336.419800
4664	65	2010-10-20	Paranormal Activity 2	3000000	84752907	177512032.0	5817.067733
4249	50	2017-01-20	Split	5000000	138141585	278964806.0	5479.296120
5189	90	2014-03-21	Godâ□□s Not Dead	1150000	60755732	63777092.0	5445.834087
4248	49	2017-02-24	Get Out	5000000	176040665	255367951.0	5007.359020
3517	18	2012-05-25	Les Intouchables	10800000	13182281	484873045.0	4389.565231
5063	64	2016-10-21	Moonlight	1500000	27854931	65245512.0	4249.700800

Visualizing

In [120…

```
# Plot the top 20 films with highest ROIs
f, ax = plt.subplots(figsize = (9, 6))
top_rois_plot = sns.barplot(data = sorted_ROI.head(10),
                            x = 'film_roi',
                            y = 'movie',
                            palette = 'cividis')

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

```
c:\Users\rosew\anaconda3\envs\learn-env\lib\site-packages\matplotlib\backends\backend_ag
g.py:238: RuntimeWarning: Glyph 128 missing from current font.
  font.set_text(s, 0.0, flags=flags)
c:\Users\rosew\anaconda3\envs\learn-env\lib\site-packages\matplotlib\backends\backend_ag
g.py:238: RuntimeWarning: Glyph 153 missing from current font.
  font.set_text(s, 0.0, flags=flags)
c:\Users\rosew\anaconda3\envs\learn-env\lib\site-packages\matplotlib\backends\backend_ag
g.py:201: RuntimeWarning: Glyph 128 missing from current font.
  font.set_text(s, 0, flags=flags)
c:\Users\rosew\anaconda3\envs\learn-env\lib\site-packages\matplotlib\backends\backend_ag
g.py:201: RuntimeWarning: Glyph 153 missing from current font.
  font.set_text(s, 0, flags=flags)
```



Question 2 Conclusion

Which films made the most money?

To get to a conclusion, we looked at the tn_movie_budgets dataset and narrowed our analysis to films releases from 2010 to 2019, in line with the analysis done in question one.

The key question was whether there was a relationship between the movie's production budget and its gross earnings.

Using Seaborn replots it was easy to see the relationship

> In the `Production Budget and Domestic Gross Plot`, as the production budget increases, so does the domestic gross. Most of the movies with budgets of less than 100 million dollars do not make more tham 500 million dollars. Movies that made more tham 600 million dollars had production budgets of over 200 million. Also, the correlation coeeficient between the production budget and domestic gross is very strong strong (0.73).
>
> In the `Production Budget and Worldwide Gross` plot we can see a similar trend. The correlation coefficient between the two variables was very strong (0.8).
>
>> The conclusion is that there is a very strong relationship between production budget and gross earnings.

We further investigated the ROI. By calculating the ROI using worldwide gross, we saw that many movies had negative returns.

However, there were many movies that made 50x or even 400x their production budget. Looking at the top 20 movies with the highest ROIs, we can see that most of these are horror films.

RECOMMENDATIONS

Based on these findings, this is the question to ask ourselves:

- How big is our production budget? We can merge our answer with that of question one.

The conclusion is that is we have a lot of money to invest, we can choose a sci-fi/fantasy/superhero franchise film, use a large production budget, and make more gross earnings.

With a small budget, we can choose a horror film and get a large ROI.

Question 3: When should the films be released?

Which release months make the most money? (We will use The Number Movie Budgets Data)

In [120…
```
# We will use the budge and earnings table and compare release month to worldwide and do
budget_and_earnings.head(10)
```

Out[1208]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	film_roi
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1.045664e+09	154.667286
2	3	2019-06-07	Dark Phoenix	350000000	42762350	1.497624e+08	-57.210757
3	4	2015-05-01	Avengers: Age of Ultron	330600000	459005868	1.403014e+09	324.384139
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1.316722e+09	315.369636
5	6	2015-12-18	Star Wars Ep. VII: The Force Awakens	306000000	936662225	2.053311e+09	571.016739
6	7	2018-04-27	Avengers: Infinity War	300000000	678815482	2.048134e+09	582.711400
8	9	2017-11-17	Justice League	300000000	229024295	6.559452e+08	118.648403
9	10	2015-11-06	Spectre	300000000	200074175	8.796209e+08	193.206974
10	11	2012-07-20	The Dark Knight Rises	275000000	448139099	1.084439e+09	294.341491
11	12	2018-05-25	Solo: A Star Wars Story	275000000	213767512	3.931513e+08	42.964126

In [120…
```
# We had cleaned the table, but let's check whether there are null values
budget_and_earnings.isna().sum()
# There are no null values
```

Out[1209]:
```
id                   0
release_date         0
movie                0
production_budget    0
domestic_gross       0
worldwide_gross      0
film_roi             0
dtype: int64
```

In [121…
```
# Checking whether there are duplicates
budget_and_earnings_duplicates = budget_and_earnings[budget_and_earnings.duplicated()]
```

```python
print(len(mb_duplicates))
# There are no duplicates
```

0

In [121…

```python
# Our columns of interest are release date (month)
# We can isolate these by dropping the columns we don't need for better readability
# Dropping the production budget column
movie_releases = budget_and_earnings.drop('production_budget', axis = 1)
# Dropping the movie title column
movie_releases = movie_releases.drop('movie', axis = 1)
# Dropping the film roi column
movie_releases = movie_releases.drop('film_roi', axis = 1)
movie_releases.head()
```

Out[1211]:

	id	release_date	domestic_gross	worldwide_gross
1	2	2011-05-20	241063875	1.045664e+09
2	3	2019-06-07	42762350	1.497624e+08
3	4	2015-05-01	459005868	1.403014e+09
4	5	2017-12-15	620181382	1.316722e+09
5	6	2015-12-18	936662225	2.053311e+09

In [121…

```python
# Quick preview of release dates with highest worldwide gross
movie_releases.groupby('release_date')['worldwide_gross'].sum().sort_values(ascending =
```

Out[1212]:

```
release_date
2015-12-18    2.418556e+09
2013-11-22    2.306838e+09
2018-04-27    2.048134e+09
2018-12-21    1.688254e+09
2012-05-04    1.664107e+09
                  ...
2015-01-01    0.000000e+00
2011-05-10    0.000000e+00
2015-11-10    0.000000e+00
2010-10-05    0.000000e+00
2019-11-22    0.000000e+00
Name: worldwide_gross, Length: 726, dtype: float64
```

In [121…

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

Out[1213]:

```
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: int32
```

In [121…

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

In [121…

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

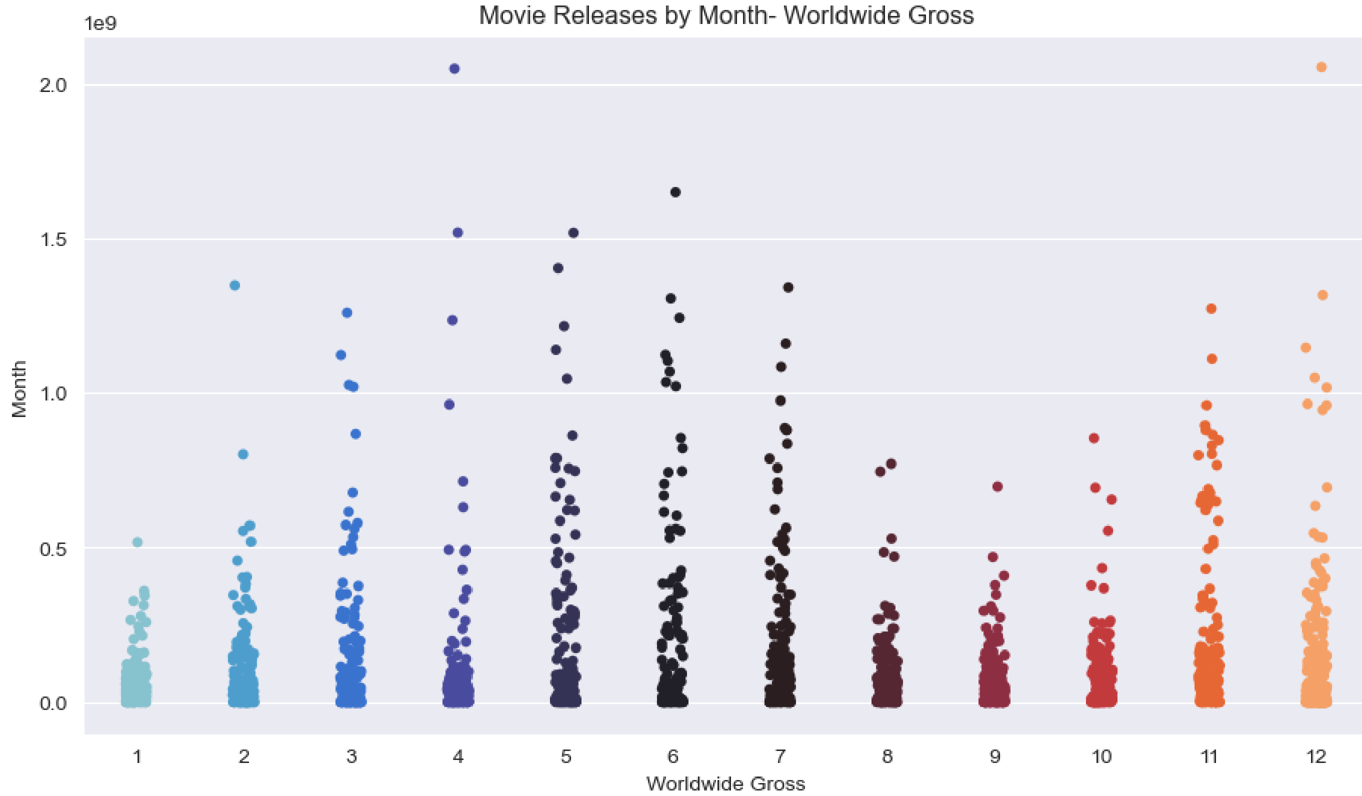
In [121…
```
movie_releases.sort_values(by = 'worldwide_gross', ascending = False).head(15)
```

Out[1216]:

	id	release_date	domestic_gross	worldwide_gross	day	month
5	6	2015-12-18	936662225	2.053311e+09	Friday	12
6	7	2018-04-27	678815482	2.048134e+09	Friday	4
33	34	2015-06-12	652270625	1.648855e+09	Friday	6
66	67	2015-04-03	353007020	1.518723e+09	Friday	4
26	27	2012-05-04	623279547	1.517936e+09	Friday	5
3	4	2015-05-01	459005868	1.403014e+09	Friday	5
41	42	2018-02-16	700059566	1.348258e+09	Friday	2
260	61	2011-07-15	381193157	1.341693e+09	Friday	7
4	5	2017-12-15	620181382	1.316722e+09	Friday	12
112	13	2018-06-22	417719760	1.305773e+09	Friday	6
155	56	2013-11-22	400738009	1.272470e+09	Friday	11
134	35	2017-03-17	504014165	1.259200e+09	Friday	3
43	44	2018-06-15	608581744	1.242521e+09	Friday	6
22	23	2017-04-14	225764765	1.234846e+09	Friday	4
47	48	2013-05-03	408992272	1.215392e+09	Friday	5

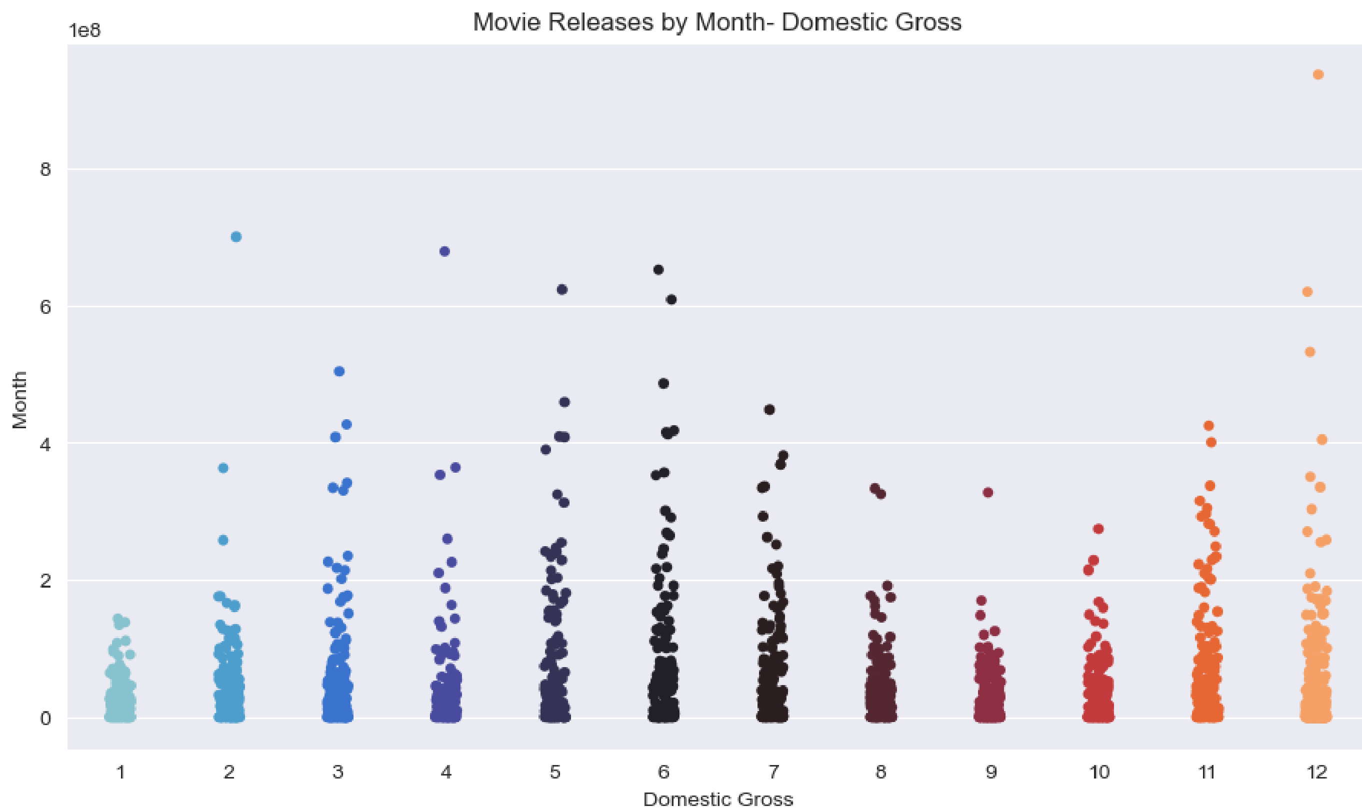
Visualizing

In [121…
```
# Plot to see the movies that were released at different months- Worldwide Gross

f, ax = plt.subplots(figsize = (11, 6))
movies = sns.stripplot(data = movie_releases,
                   x = "month",
                   y = "worldwide_gross",
                    palette = 'icefire')
movies.set_title('Movie Releases by Month- Worldwide Gross')
movies.set_xlabel('Worldwide Gross')
movies.set_ylabel('Month');
```



In [121…

```
# Plot to see the movies that were released at different months- Domestic Gross

f, ax = plt.subplots(figsize = (11, 6))
movies2 = sns.stripplot(data = movie_releases,
                        x = "month",
                        y = "domestic_gross",
                         palette = 'icefire')
movies2.set_title('Movie Releases by Month- Domestic Gross')
movies2.set_xlabel('Domestic Gross')
movies2.set_ylabel('Month');
```

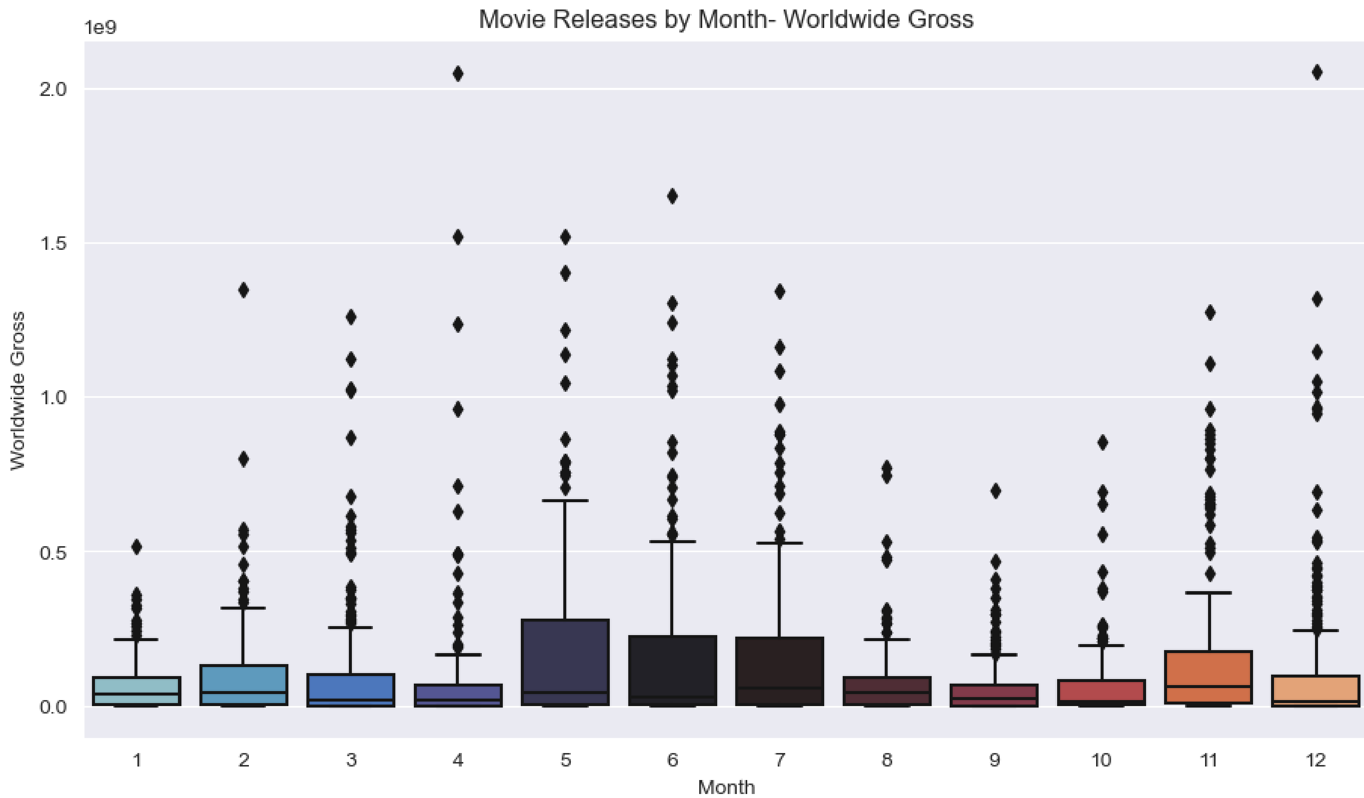


In [121…

```
# Let's use a box for another similar visualization- Worldwide Gross
```

```
f, ax = plt.subplots(figsize = (11, 6))
movies3 = sns.boxplot(data = movie_releases,
                   x = "month",
                   y = "worldwide_gross",
                    palette = 'icefire')
movies3.set_title('Movie Releases by Month- Worldwide Gross')
movies3.set_ylabel('Worldwide Gross')
movies3.set_xlabel('Month')
;
```

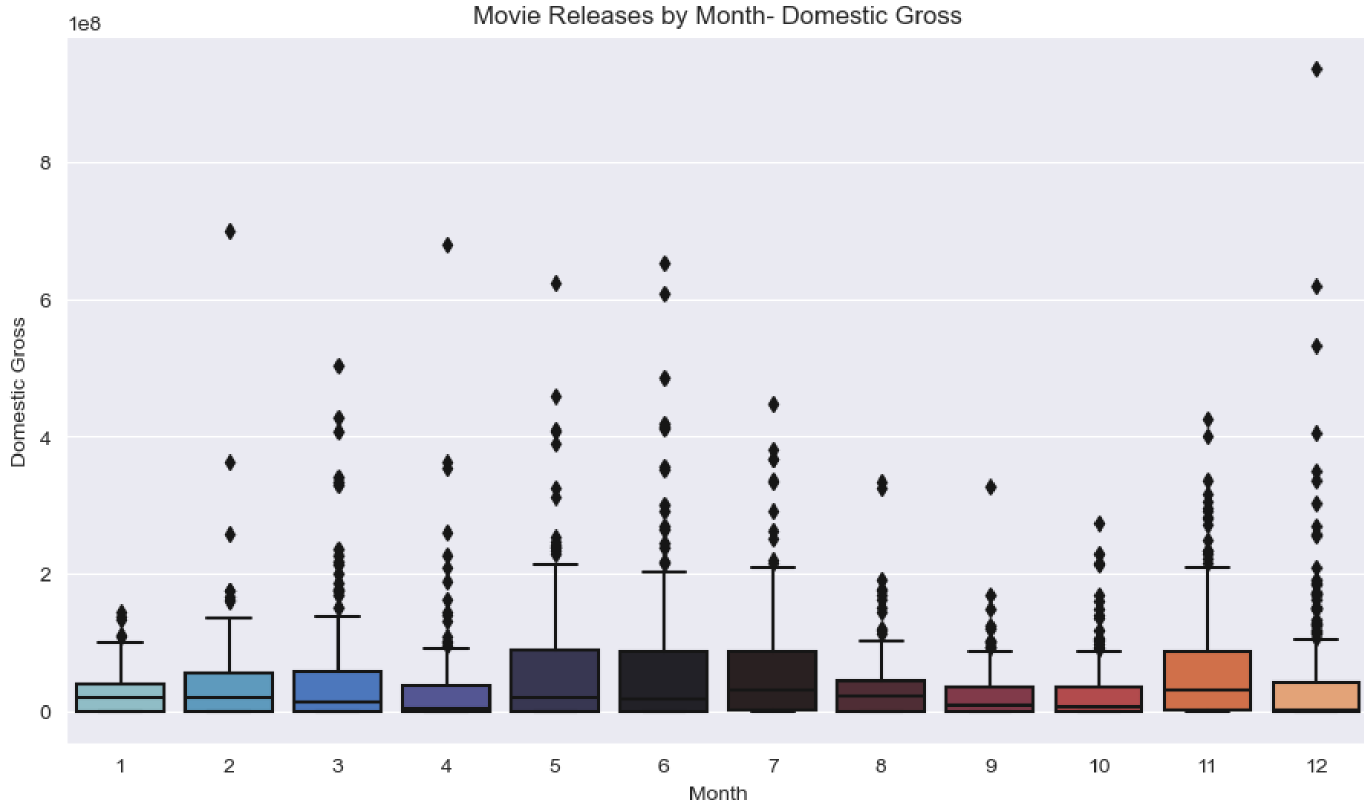
Out[1219]: ''



In [122…

```
# Let's use a box for another similar visualization- Domestic Gross
f, ax = plt.subplots(figsize = (11, 6))
movies4 = sns.boxplot(data = movie_releases,
                   x = "month",
                   y = "domestic_gross",
                    palette = 'icefire')
movies4.set_title('Movie Releases by Month- Domestic Gross')
movies4.set_ylabel('Domestic Gross')
movies4.set_xlabel('Month')
;
```

Out[1220]: ''



In [122…
```
# Group movie releases by month and find mean and median worldwide gross for each month
movie_releases_by_month = movie_releases.groupby('month')['worldwide_gross'].agg(['mean'
```

In [122…
```
# Group movie releases by month and find mean and median domestic gross for each month
movie_releases_by_month_domestic = movie_releases.groupby('month')['domestic_gross'].agg
```

In [122…
```
# Sort by median, since that number is less affected by outliers- Worldwide Gross
movie_releases_by_month.sort_values(by = 'median', ascending = False)
```

Out[1223]:

month	mean	median
11	1.726833e+08	60217171.0
7	1.735722e+08	57273049.0
2	9.961363e+07	43528634.0
5	1.864977e+08	43061376.0
8	7.585777e+07	40650842.0
1	6.586246e+07	36285960.5
6	1.774432e+08	29867459.5
9	5.853239e+07	22281732.0
3	1.082374e+08	20592763.0
4	8.657684e+07	17478366.5
12	1.052797e+08	13779342.5
10	6.583288e+07	12429202.0

In [122…
```
# Sort by median, since that number is less affected by outliers- Domestic Gross
movie_releases_by_month_domestic.sort_values(by = 'median', ascending = False)
```

Out[1224]:

	mean	median

month		
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

Question 3 Conclusion

To answer this question, the tn_movie_budgets database provided movie release dates and gross worldwide and domestic earnings.

Looking at the domestic gross by month over 10 years (2010 - 2019), we can find the months with the highest domestic gross.

Using these ten years gives data with ten occurences for each month.

The median was used as an indicator to rule out outliers.

The five highest grossing months based on domestic returns are:

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

The five highest grossing months based on worldwide returns are:

- November - $60,217,171
- July - $57,273,049
- February- $43,528,634
- January - $43,061,376
- August - $40,650,842

RECOMMENDATIONS

Based on these findings, the common months are July, November, August, and February. This is also evidenced from the bar and strip plots where the figures for these months are larger than the rest.

Question 4: What is the average rating per genre:

We will use the IMDB Databse, and based on the image of the data, we know to use the basics and ratings tables.

Here is the image:

example

In [122…

```
# Checking for the columns in the movie_basics table
basics = """
SELECT *
FROM movie_basics
;
"""
pd.read_sql(basics, conn).head()
```

Out[1225]:

	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

In [122…

```
# Checking for the columns in the movie_ratings table
ratings = """
SELECT *
FROM movie_ratings
;
"""
pd.read_sql(ratings, conn).head()
```

Out[1226]:

	movie_id	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 [122…

```
# Joining the movie_basics and movie_ratings tables from the IMDB database
# Using inner join to only get the films both tables have in common
sql_query = pd.read_sql_query(
    """
    SELECT movie_id, averagerating, numvotes, genres, start_year, primary_title
    FROM movie_basics
    INNER JOIN movie_ratings
```

```
    USING(movie_id)
    """,
    conn,
)

df = pd.DataFrame(sql_query, columns=["movie_id", "genres", 'averagerating', 'numvotes',

print(df)
```

```
        movie_id                 genres  averagerating  numvotes  start_year  \
0      tt0063540     Action,Crime,Drama            7.0        77        2013
1      tt0066787        Biography,Drama            7.2        43        2019
2      tt0069049                  Drama            6.9      4517        2018
3      tt0069204           Comedy,Drama            6.1        13        2018
4      tt0100275   Comedy,Drama,Fantasy            6.5       119        2017
...          ...                    ...            ...       ...         ...
73851  tt9913084            Documentary            6.2         6        2019
73852  tt9914286           Drama,Family            8.7       136        2019
73853  tt9914642            Documentary            8.5         8        2017
73854  tt9914942                   None            6.6         5        2019
73855  tt9916160            Documentary            6.5        11        2019

                         primary_title
0                            Sunghursh
1      One Day Before the Rainy Season
2           The Other Side of the Wind
3                      Sabse Bada Sukh
4             The Wandering Soap Opera
...                                ...
73851                Diabolik sono io
73852                Sokagin Çocuklari
73853                        Albatross
73854        La vida sense la Sara Amat
73855                       Drømmeland

[73856 rows x 6 columns]
```

In [122…
```
# Looking at the number of rows with null values
df.isna().sum()
```

Out[1228]:
```
movie_id           0
genres           804
averagerating      0
numvotes           0
start_year         0
primary_title      0
dtype: int64
```

In [122…
```
# Dropping the null values in the genres column
joined_imdb = df.dropna()
```

In [123…
```
# Cheking whether it worked
joined_imdb.isna().sum()
```

Out[1230]:
```
movie_id         0
genres           0
averagerating    0
numvotes         0
start_year       0
primary_title    0
dtype: int64
```

In [123…
```
# Dropping any duplicates
joined_imdb.drop_duplicates()
```

Out[1231]:

	movie_id	genres	averagerating	numvotes	start_year	primary_title

0	tt0063540	Action,Crime,Drama	7.0	77	2013	Sunghursh
1	tt0066787	Biography,Drama	7.2	43	2019	One Day Before the Rainy Season
2	tt0069049	Drama	6.9	4517	2018	The Other Side of the Wind
3	tt0069204	Comedy,Drama	6.1	13	2018	Sabse Bada Sukh
4	tt0100275	Comedy,Drama,Fantasy	6.5	119	2017	The Wandering Soap Opera
...	...	...	...	...	...	...
73850	tt9913056	Documentary	6.2	5	2019	Swarm Season
73851	tt9913084	Documentary	6.2	6	2019	Diabolik sono io
73852	tt9914286	Drama,Family	8.7	136	2019	Sokagin Çocuklari
73853	tt9914642	Documentary	8.5	8	2017	Albatross
73855	tt9916160	Documentary	6.5	11	2019	Drømmeland

73052 rows × 6 columns

In [123…

```
# Looking at the data types to ensure we have numbers for rating and number of votes
joined_imdb.dtypes
```

Out[1232]:

```
movie_id           object
genres             object
averagerating     float64
numvotes            int64
start_year          int64
primary_title      object
dtype: object
```

In [123…

```
# Make sure the years are within the 10 year scope
joined_imdb['start_year'].unique()
# Yes, they are
```

Out[1233]:

```
array([2013, 2019, 2018, 2017, 2010, 2011, 2012, 2015, 2016, 2014],
      dtype=int64)
```

Deciding on the threshold for number of votes

It is important to note that the films that have a small number of votes have the potential to skew our data results.

For example, a film could only have 10 votes but idf these are good votes, the average rating will be very high.

However, a big box office film could have over a million reviews ranging from low to high, but this is a normalized average rating.

The approach is to check the highest and lowest number of votes and then plot to see th distribution.

We will then rule out any films with less than 100,000 votes to give a normalized average rating and avoid films with few votes that could skew the results.

In [123…

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

Out[1234]:

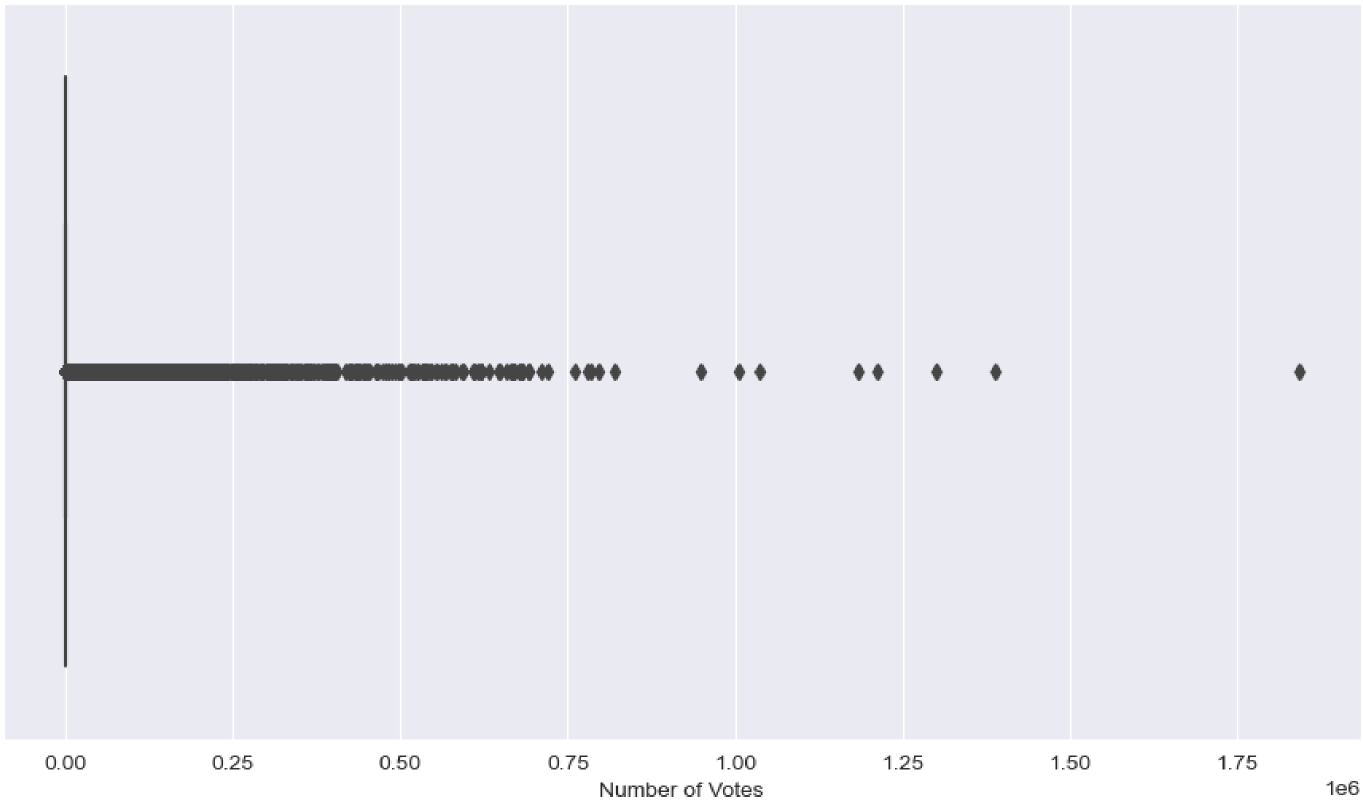
```
325      1183655
12072    1211405
280      1299334
```

```
2241     1387769
2387     1841066
Name: numvotes, dtype: int64
```

Visualizing

In [123…

```
# See distribution of numvotes
# Most of them are close to zero and we will remove these
fig, ax = plt.subplots(figsize = (11, 6))
joined_imdb['numvotes'].sort_values()
plot = sns.boxplot(data=joined_imdb['numvotes'], x=joined_imdb['numvotes'], palette = "r
plot.set_xlabel('Number of Votes');
```



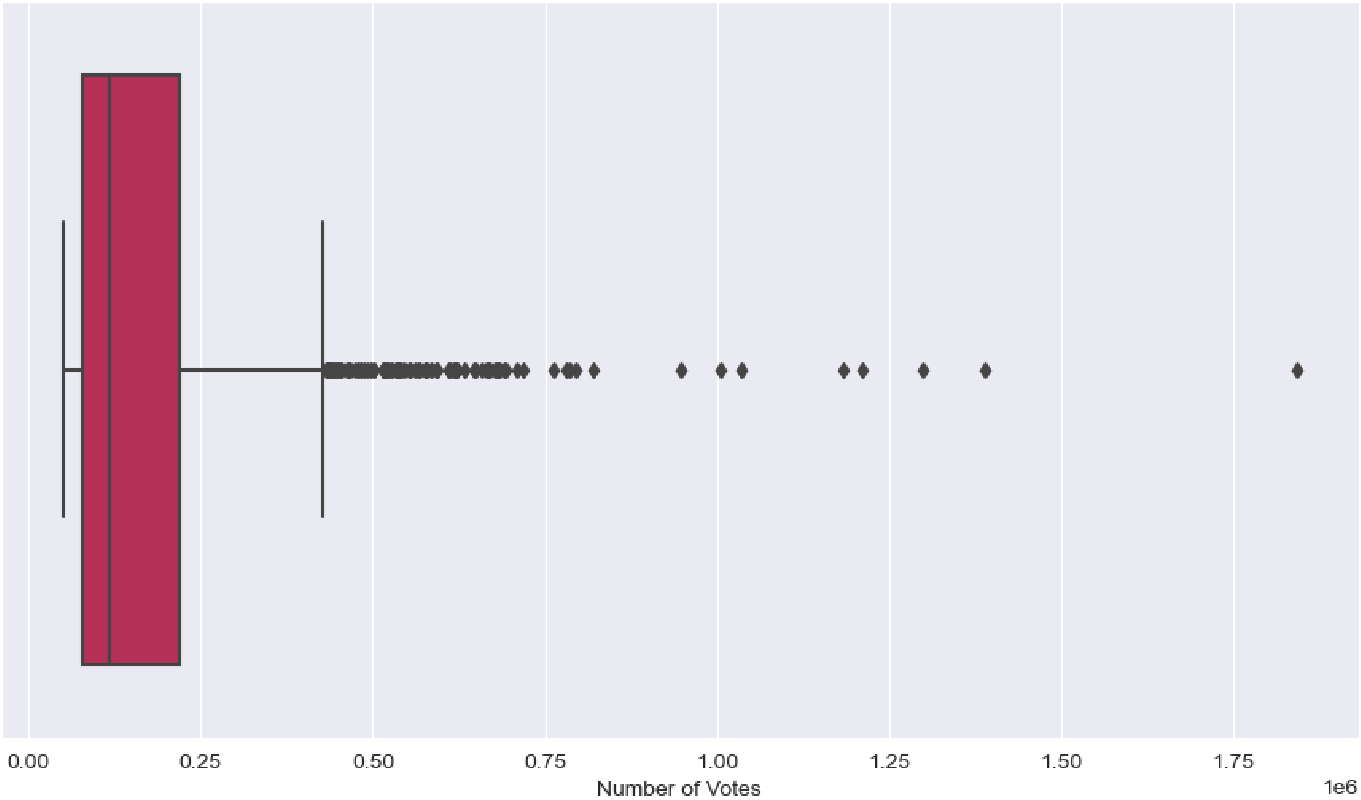
In [123…

```
# Let's only include films with over 100,000 votes so that the movies with very few rati
new_imdb = joined_imdb.loc[joined_imdb['numvotes'] > 50000]
print(new_imdb.shape)
print(joined_imdb.shape)
```

```
(1064, 6)
(73052, 6)
```

In [123…

```
# Check the distribution again
fig2, ax = plt.subplots(figsize = (11, 6))
new_imdb['numvotes'].sort_values()
plot2 = sns.boxplot(data=new_imdb['numvotes'], x=new_imdb['numvotes'], palette = "rocket
plot2.set_xlabel('Number of Votes');
```



Dealing with the genres column

We notice that there are multiple genres listed for each film, all in the same column.
This calls for the need to flatten the DF.

However, this will duplicate the films, but there will be more than one genre for each value witha correpsonding rating.

Even though the ratings will be available more than ince for different ratings, this is fine since our goal is to find the rating for each genre.

We will end up with statistics for each genre.

In [123…

```
new_imdb.head()
```

Out[1238]:

	movie_id	genres	averagerating	numvotes	start_year	primary_title
47	tt0359950	Adventure,Comedy,Drama	7.3	275300	2013	The Secret Life of Walter Mitty
51	tt0365907	Action,Crime,Drama	6.5	105116	2014	A Walk Among the Tombstones
52	tt0369610	Action,Adventure,Sci-Fi	7.0	539338	2015	Jurassic World
54	tt0376136	Comedy,Drama	6.2	94787	2011	The Rum Diary
61	tt0398286	Adventure,Animation,Comedy	7.8	366366	2010	Tangled

In [123…

```
# Let's see which movies have the highest ratings after we've adjusted for only adequate
highestratings = new_imdb.sort_values(by = 'averagerating', ascending = False)
highestratings.head(10)
```

Out[1239]:

	movie_id	genres	averagerating	numvotes	start_year	primary_title
56850	tt5813916	Action,Drama,War	9.3	100568	2016	The Mountain II

2387	tt1375666	Action,Adventure,Sci-Fi	8.8	1841066	2010	Inception
43420	tt4154796	Action,Adventure,Sci-Fi	8.8	441135	2019	Avengers: Endgame
280	tt0816692	Adventure,Drama,Sci-Fi	8.6	1299334	2014	Interstellar
2770	tt1424432	Biography,Documentary,Sport	8.6	55318	2010	Senna
7125	tt1675434	Biography,Comedy,Drama	8.5	677343	2011	The Intouchables
26774	tt2582802	Drama,Music	8.5	616916	2014	Whiplash
50962	tt5074352	Action,Biography,Drama	8.5	123638	2016	Dangal
47436	tt4633694	Action,Adventure,Animation	8.5	210869	2018	Spider-Man: Into the Spider-Verse
43419	tt4154756	Action,Adventure,Sci-Fi	8.5	670926	2018	Avengers: Infinity War

In [124…
```
# Making the genres column values each a list, instead of one long string
highestratings['genres'] = highestratings['genres'].str.split(",")
```

In [124…
```
# Confirming
highestratings.head()
```

Out[1241]:

	movie_id	genres	averagerating	numvotes	start_year	primary_title
56850	tt5813916	[Action, Drama, War]	9.3	100568	2016	The Mountain II
2387	tt1375666	[Action, Adventure, Sci-Fi]	8.8	1841066	2010	Inception
43420	tt4154796	[Action, Adventure, Sci-Fi]	8.8	441135	2019	Avengers: Endgame
280	tt0816692	[Adventure, Drama, Sci-Fi]	8.6	1299334	2014	Interstellar
2770	tt1424432	[Biography, Documentary, Sport]	8.6	55318	2010	Senna

In [124…
```
# Flatten the DF
separated_genres = (highestratings
              .set_index(['primary_title','averagerating'])['genres']
              .apply(pd.Series)
              .stack()
              .reset_index()
              .drop('level_2', axis = 1)
              .rename(columns = {0 : 'genres'}))
```

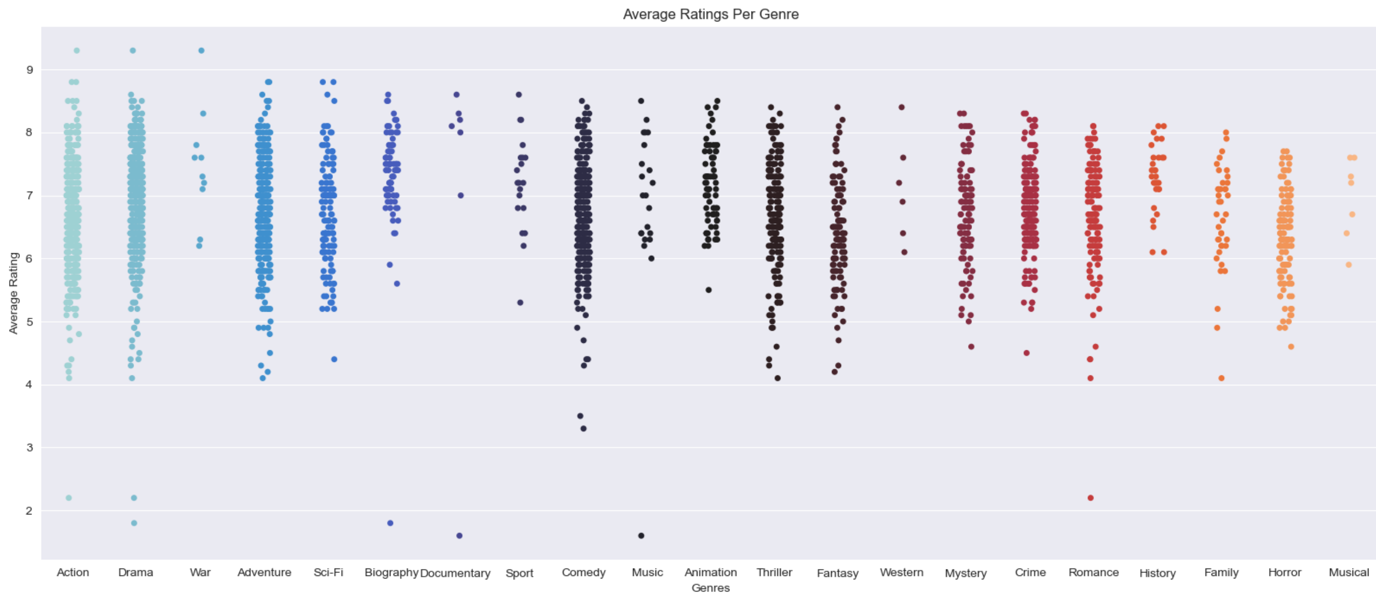
In [124…
```
separated_genres.head()
```

Out[1243]:

	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

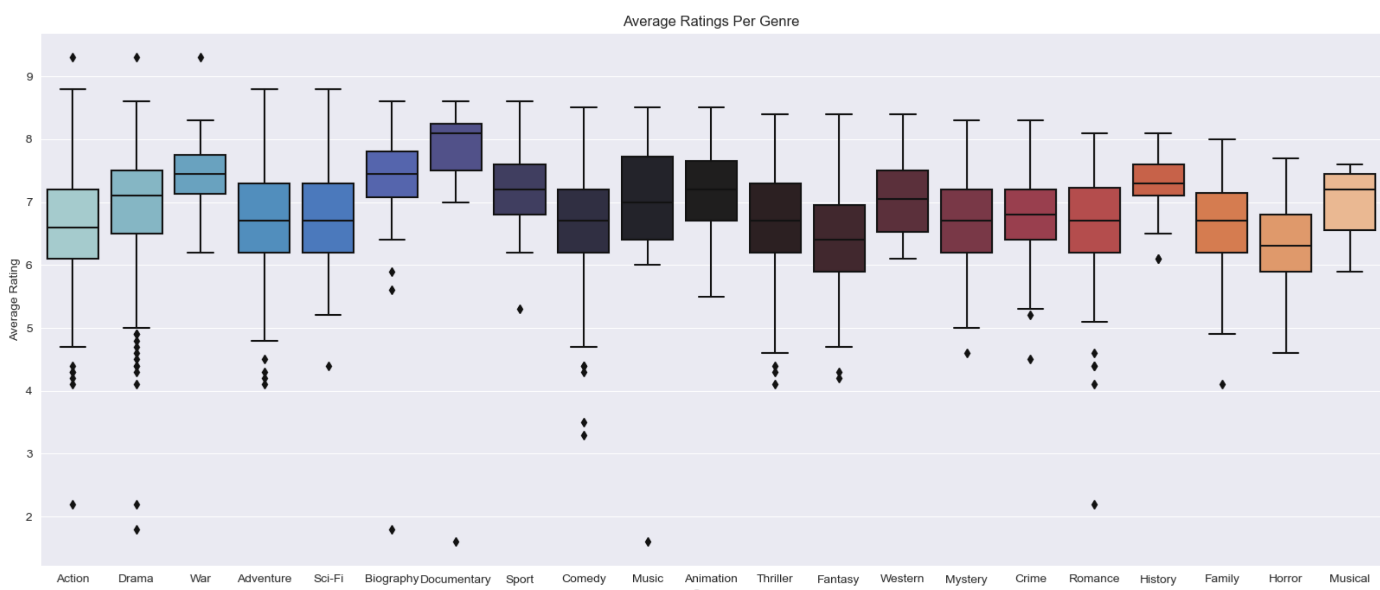
In [124…
```
# Wel'' use a strip plot to see the distribution of each genre's average ratings
f, ax = plt.subplots(figsize = (20, 8))
ratings_plot = sns.stripplot(data = separated_genres,
                   x = "genres",
                   y = "averagerating",
                    palette = 'icefire')
ratings_plot.set_title('Average Ratings Per Genre')
```

```
ratings_plot.set_ylabel('Average Rating')
ratings_plot.set_xlabel('Genres');
```



In [124…

```
# Let's use a box plot for the same
f, ax = plt.subplots(figsize = (20, 8))
ratings_plot2 = sns.boxplot(data = separated_genres,
                            x = "genres",
                            y = "averagerating",
                             palette = 'icefire')
ratings_plot2.set_title('Average Ratings Per Genre')
ratings_plot2.set_ylabel('Average Rating')
ratings_plot2.set_xlabel('Genres');
```



In [124…

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

Out[1246]:

	mean	median
genres		
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 4 Conclusion

To identify the genres that receive the highest ratings, we calculated the average ratings per genre.

This was using the IMDB database, specifically the movie basics and movie ratings tables.

After joining the two tables, we saw that the films ranged from 2010 to 2019, which is in line with the previous questions.

Ratings for movies with less than 100,000 votes were included since these would skew the results greatly (a movie with only 10 votes and a high ratings would be based on very little data)

The movie genres were then separated into single-genre categories, and a `strip plot, swarm plot, and mean and median values` were calculated.

The plots show us the distribution of each genre's ratings.

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 rating" of 7.45. And tied for second highest is "War", with a median "average rating" of 7.45.

However, all of the median average ratings fall between 6.30 and 8.10.

Therefore, even though "Documentary", "Biography" and "War" movies are the highest rated movies, their ratings are not much higher than "Animation", for example, with a median "average rating" of 7.20.

RECOMMENDATION

We need to tie our recommendations with the previous question and rely on the median since this is less skewed than mean.

The genres that have been previously identified in question one as the ones with the highest grossing returns are animated, superhero, scifi, and fantasy movies.

Scifi, animated, and fantasy movies are in this analysis and affirm our previous analysis that they are great options to focus on since there is no large variation in the median results.