## 0.1 Final Project Submission

Please fill out:

- Student name: Brian Bentson
- Student pace: Full time
- Scheduled project review date/time: Monday 3/22 @ 3pm CST
- Instructor name: James Irving
- Blog post URL:

## 0.2 Functions

In [351]:

```
#import relevant modules
import os, glob
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.style as style
import matplotlib.ticker as tick
plt.style.use('grayscale')
import warnings
warnings.filterwarnings('ignore')
```

In [352]:

```
pd.set_option('display.max_rows', 100)
```

### 0.2.1 Function to Explore Table and Column Info

In [353]:
```
def get_info(table_name, column=None):
    if column == None:
        print(f'Table Name: {table_name}')
        print('\n')
        print('Table Columns')
        print(tables[table_name].columns)
        print('\n')
        print('Table Info')
        print(tables[table_name].info())
        print('\n')
        print('Table Descriptive Statistics')
        print(tables[table_name].describe())
    else:
        print(f'Table Name: {table_name}')
        print('\n')
        print('Table Columns')
        print(tables[table_name].columns)
        print('\n')
        print('Table Info')
        print(tables[table_name].info())
        print('\n')
        print(f'{column.title()} Descriptive Statistics')
        print(tables[table_name][column].describe())
        print('\n')
        print('Table Values')
        print(tables[table_name][column].value_counts(dropna=False
        print('\n')
        print('Unique Values')
        print(tables[table_name][column].unique())
```

### 0.2.2 Function to Update Tick Labels

Sourced from: [https://dfrieds.com/data-visualizations/how-format-large-tick-values.html (https://dfrieds.com/data-visualizations/how-format-large-tick-values.html)](https://dfrieds.com/data-visualizations/how-format-large-tick-values.html)

In [354]:

```
def reformat_large_tick_values(tick_val, pos):
    """
    Turns large tick values (in the billions, millions and thousar
    """
    if tick_val >= 1000000000:
        val = round(tick_val/1000000000, 1)
        new_tick_format = '${:}B'.format(val)
    elif tick_val >= 1000000:
        val = round(tick_val/1000000, 1)
        new_tick_format = '${:}M'.format(val)
    elif tick_val >= 1000:
        val = round(tick_val/1000, 1)
        new_tick_format = '${:}K'.format(val)
    elif tick_val < 1000:
        new_tick_format = round(tick_val, 1)
    else:
        new_tick_format = tick_val

    # make new_tick_format into a string value
    new_tick_format = str(new_tick_format)

    # code below will keep 4.5M as is but change values such as 4.
    index_of_decimal = new_tick_format.find(".")

    if index_of_decimal != -1:
        value_after_decimal = new_tick_format[index_of_decimal+1]
        if value_after_decimal == "0":
            # remove the 0 after the decimal point since it's not
            new_tick_format = new_tick_format[0:index_of_decimal]

    return new_tick_format
```

# 1 Business Statement

Based on the suceess of their peers, Microsoft has decided to create a new movie studio focused on creating original video content. They have no direct movie creation experience and want to leverage historical movie data in order to determine what are leading indicators of a successful movie. This analysis can be used to make data-driven decisions on parameters of a prospective first movie.

# 2 Analysis Methodology

I will be analyzing historic movie data to find actionable insights for the head of Mircrosoft's new movie studio on how to create a successful introduction to the movie industry.

A movie's success can be judged by many factors centered around financial and social measures. Since it is imperative to start on a good footing when entering a new industry, I have decided to focus my analysis on the financial aspect of measuring success. This will mean that a successful movie will have a high relative return on investment.

# 3 Data Collection

Since I am choosing to judge movie success on the financial metric of return on investment (ROI), I need to gather the correct data in order to make that calculation. The following data will be gathered:

- Movie-specific meta-data
- Production Cost
- Global Revenue

I have 11 separate files that provide movie meta-data which will be helpful in the analysis. I will import them using panda and determine which files should be utilized in the analysis

## 3.1 Import Movie Data into Pandas

### 3.1.1 Import Modules

In [355]:

```
#import relevant modules
import os, glob
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

### 3.1.2 Preview All Files

In [356]:

```
#function to preview all available files
files = glob.glob(f'../dsc-phase-1-project/zippedData/*.[c,t]sv*')

tables = {}
dashes = '---'*25
```

```
dashes = '-'*25

for file in files:
    if 'csv' in file:
        table_name = file.replace('.csv.gz','').split('/')[-1].rep
        tables[table_name] = pd.read_csv(file)
        print(dashes)
        print(f'Table Name: {table_name}')
        display(tables[table_name].head())
    else:
        table_name = file.replace('.tsv.gz','').split('/')[-1].rep
        tables[table_name] = pd.read_csv(file, delimiter='\t', enc
        print(dashes)
        print(f'Table Name: {table_name}')
        display(tables[table_name].head())

rt_reviews = tables['rt_reviews']
rt_movie_info = tables['rt_movie_info']
tmdb_movies = tables['tmdb_movies']
tn_movie_budgets = tables['tn_movie_budgets']
imdb_title_basics = tables['imdb_title_basics']
imdb_title_ratings = tables['imdb_title_ratings']
imdb_name_basics = tables['imdb_name_basics']
imdb_title_principals = tables ['imdb_title_principals']
imdb_title_crew = tables['imdb_title_crew']
imdb_title_akas = tables['imdb_title_akas']
bom_movie_gross = tables['bom_movie_gross']
```

```
------------------------------------------------------------------
--------
Table Name: imdb_title_crew
```

| | tconst | directors | writers |
|---|---|---|---|
| **0** | tt0285252 | nm0899854 | nm0899854 |
| **1** | tt0438973 | NaN | nm0175726,nm1802864 |
| **2** | tt0462036 | nm1940585 | nm1940585 |
| **3** | tt0835418 | nm0151540 | nm0310087,nm0841532 |
| **4** | tt0878654 | nm0089502,nm2291498,nm2292011 | nm0284943 |

```
------------------------------------------------------------------
--------
Table Name: tmdb_movies
```

**Analysis**
It looks like the most relevant files to be analyzed are as follows:

- **imdb_title_basics**
  - Has tconst for linking to other files
  - Has genres for genre specific analysis
- **imdb_title_akas**
  - Has is_original_title to help filter duplicate movie titles from imdb_title_basics
- **tn_movie_budgets**
  - Has production_budget, gross domestic and worldwide revenue
- **imdb_title_ratings**
  - Has user ratings
- **bom_movie_gross**
  - Has movie studios

# 4 Data Cleaning

## 4.1 Understanding Raw Data

In order to determine if the right information is present and how to join different tables together for my analysis, I first need to understand what each piece of data is and how it can be used. I will do some short data exploration to understand the data better and decide which data processing techniques to use.

### 4.1.1 Column Meanings for Each Table

**imdb_title_crew**

- **tconst**: Unique identifier for each movie (PRIMARY KEY)
- **directors**: Director code
- **writers**: Writer code

**tmdb_movies**

- Unnamed: 0: Can be removed or set as index
- genre_id's: Genre code
- id: Unknown
- original_language: movie language

- original_title: movie title
- popularity: Unknown
- release_date: movie release date
- title: movie title
- vote_average: Unknown
- vote_count: Number of votes

**imdb_title_akas**

- title_id: movie id
- ordering: Unknown
- title: movie title
- region: Country of origin
- language: movie language
- types: Unknown
- attributes: Unknown
- is_original_title: Unknown

**imdb_title_ratings**

- tconst: Unknown
- **averagerating**: movie rating
- **numvotes**: Number of votes

**imdb_name_basics**

- **nconst**: Unique identifier for person (PRIMARY KEY)
- **primary_name**: Name
- birth_year: Year born
- death_year: Year died
- **primary_profession**: Job Roles
- known_for_titles: title id's

**rt_reviews**

- id: Unknown
- review: Review comments
- rating: Movie rating
- fresh: fresh or rotten score
- critic: Critic Name
- top_critic: Unknown

- publisher: Publisher Name
- date: Unknown

**imdb_title_basics**

- **tconst**: Unique identifier for movie
- **primary_title**: Common Movie Name
- original_title: Native Movie Name
- **start_year**: Year of release
- runtime_minutes: Movie length in minutes
- **genres**: movie genre

**rt_movie_info**

- id: Unknown
- synopsis: Movie synopsis
- rating: movie parental rating
- genre: Movie genre
- director: Movie director
- writer: Movie writer
- theater_date: Theater release data
- dvd_date: DVD release date
- currency: Currency type
- box_office: Unknown
- run_time: Movie length
- studio: Movie Production Studio

**tn_movie_budgets**

- id: Unknown
- release_date: Movie release date
- movie: Movie title (PRIMARY KEY)
- **production_budget**: Movie production budget in USD
- **domestic_gross**: Gross revenue domestically
- **worldwide_gross**: Gross revenue worldwide

**bom_movie_gross**

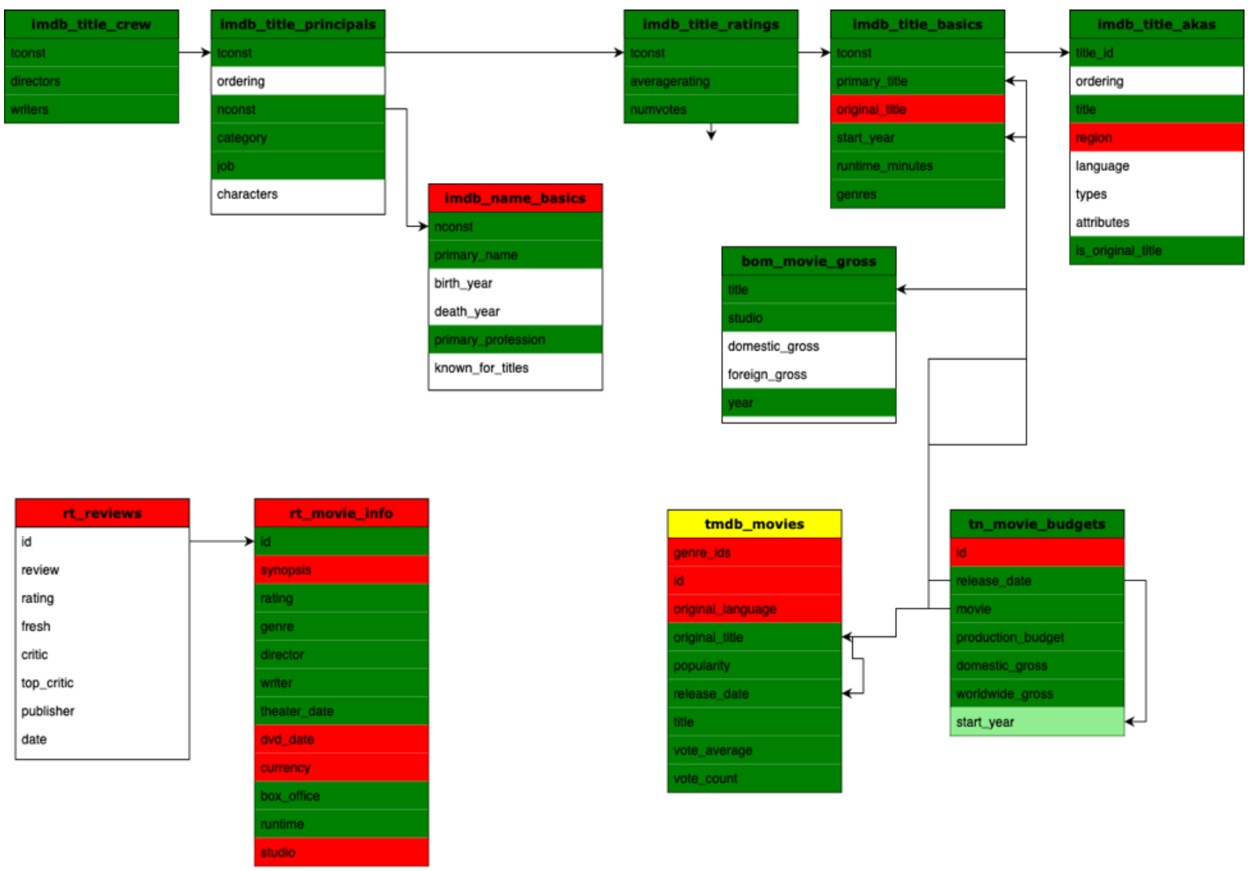
- title: Movie title (PRIMARY KEY)
- studio: Movie studio
- **domestic_gross**: Gross revenue domestically

- **foreign_gross**: Gross revenue worldwide
- year: Release year (PRIMARY KEY)

**imdb_title_principals**

- tconst: Unique identifier for movie (PRIMARY KEY)
- ordering: Unknown
- nconst: Unique identifier for person (PRIMARY KEY)
- category: Job role
- job: Unknown
- characters:Character played in movie

### 4.1.2 Entity Relationship Diagram




## 4.2 Clean Up Tables for Joins

### 4.2.1 tn_movie_budgets

In [357]:
```
#number of rows, columns and first 5 rows
display(tn_movie_budgets.shape,tn_movie_budgets.head())
```

(5782, 6)

| | 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 |

#### 4.2.1.1 Remove Columns

In [358]:
```
#removing id because it doesn't like up with any other id's in oth
tn_movie_budgets = tn_movie_budgets.drop('id',axis=1)
tn_movie_budgets.head()
```

Out[358]:

| | release_date | movie | production_budget | domestic_gross | worldwide_gross |
|---|---|---|---|---|---|
| 0 | Dec 18, 2009 | Avatar | $425,000,000 | $760,507,625 | $2,776,345,279 |
| 1 | May 20, 2011 | Pirates of the Caribbean: On Stranger Tides | $410,600,000 | $241,063,875 | $1,045,663,875 |
| 2 | Jun 7, 2019 | Dark Phoenix | $350,000,000 | $42,762,350 | $149,762,350 |
| 3 | May 1, 2015 | Avengers: Age of Ultron | $330,600,000 | $459,005,868 | $1,403,013,963 |
| 4 | Dec 15, 2017 | Star Wars Ep. VIII: The Last Jedi | $317,000,000 | $620,181,382 | $1,316,721,747 |

#### 4.2.1.2 Convert Values

In [359]:

```
#converting release_date to datetime object to be able to extract
tn_movie_budgets['release_date'] = pd.to_datetime(tn_movie_budgets
tn_movie_budgets['start_year'] = tn_movie_budgets['release_date'].
tn_movie_budgets.head()
```

Out[359]:

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

In [360]:
```
#convert financial fields into integers for future calculations
tn_movie_budgets['production_budget'] = tn_movie_budgets['producti
tn_movie_budgets['domestic_gross'] = tn_movie_budgets['domestic_gr
tn_movie_budgets['worldwide_gross'] = tn_movie_budgets['worldwide_
tn_movie_budgets.head()
```

Out[360]:

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

#### 4.2.1.3 Check for Duplicates

In [361]:
```
#check for any duplicates for the combination of movie and release
#those are going to the be the primary keys for future joins
tn_movie_budgets.loc[tn_movie_budgets.duplicated(subset=['movie','
```

Out[361]:

| | release_date | movie | production_budget | domestic_gross | worldwide_gross | start_year |
|---|---|---|---|---|---|---|
| **3455** | 2009-06-05 | Home | 12000000 | 0 | 0 | 2009 |
| **5459** | 2009-04-23 | Home | 500000 | 15433 | 44793168 | 2009 |

1 duplicate found. Both will be removed during innner join with imdb_title_basics because they are prior to 2010.

#### 4.2.1.4 Final Table View

In [362]:
```
tn_movie_budgets
```

Out[362]:

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

### 4.2.2 imdb_title_basics

In [363]:
```
#number of rows, columns and first 5 rows
display(imdb_title_basics.shape,imdb_title_basics.head())
```

(146144, 6)

| | 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 |

#### 4.2.2.1 Remove Columns

In [364]:
```
#remove original_title because primary_title looks to be most accu
imdb_title_basics = imdb_title_basics.drop('original_title',axis=1
imdb_title_basics.head()
```

Out[364]:

| | tconst | primary_title | start_year | runtime_minutes | genres |
|---|---|---|---|---|---|
| **0** | tt0063540 | Sunghursh | 2013 | 175.0 | Action,Crime,Drama |
| **1** | tt0066787 | One Day Before the Rainy Season | 2019 | 114.0 | Biography,Drama |
| **2** | tt0069049 | The Other Side of the Wind | 2018 | 122.0 | Drama |
| **3** | tt0069204 | Sabse Bada Sukh | 2018 | NaN | Comedy,Drama |
| **4** | tt0100275 | The Wandering Soap Opera | 2017 | 80.0 | Comedy,Drama,Fantasy |

#### 4.2.2.2 Check for Duplicates

In [365]:
```
#check for any duplicates for the combination of primary_title and
imdb_title_basics.loc[imdb_title_basics.duplicated(subset=['primar
```

Out[365]:

| | tconst | primary_title | start_year | runtime_minutes | genres |
|---|---|---|---|---|---|
| **21** | tt0250404 | Godfather | 2012 | NaN | Crime,Drama |
| **117** | tt0443465 | Before We Go | 2014 | 95.0 | Comedy,Drama,Romance |
| **133** | tt0452664 | Party Crashers | 2012 | 88.0 | Comedy |
| **211** | tt0490075 | Aftermath | 2013 | 84.0 | Crime,Thriller |
| **276** | tt0800054 | The Guardians | 2010 | 88.0 | Comedy,Family |
| **...** | ... | ... | ... | ... | ... |
| **145919** | tt9886934 | The Projectionist | 2019 | 81.0 | Documentary |
| **145937** | tt9889072 | The Promise | 2017 | NaN | Drama |
| **146068** | tt9905256 | The Cross | 2012 | NaN | Thriller |
| **146119** | tt9913594 | Bacchanalia | 2017 | 72.0 | Drama,Mystery,Thriller |
| **146120** | tt9913936 | Paradise | 2019 | NaN | Crime,Drama |

3942 rows × 5 columns

Because there are almost 4000 duplicates, I will use the imdb_title_akas table to remove duplicates by ensuring imdb_title_basics holds only original titles

#### 4.2.2.3 Final Table View

In [366]:
```
imdb_title_basics
```

Out[366]:

| | tconst | primary_title | start_year | runtime_minutes | genres |
|---|---|---|---|---|---|
| **0** | tt0063540 | Sunghursh | 2013 | 175.0 | Action,Crime,Drama |
| **1** | tt0066787 | One Day Before the Rainy Season | 2019 | 114.0 | Biography,Drama |
| **2** | tt0069049 | The Other Side of the Wind | 2018 | 122.0 | Drama |
| **3** | tt0069204 | Sabse Bada Sukh | 2018 | NaN | Comedy,Drama |
| **4** | tt0100275 | The Wandering Soap Opera | 2017 | 80.0 | Comedy,Drama,Fantasy |
| **...** | ... | ... | ... | ... | ... |
| **146139** | tt9916538 | Kuambil Lagi Hatiku | 2019 | 123.0 | Drama |
| **146140** | tt9916622 | Rodolpho Teóphilo - O Legado de um Pioneiro | 2015 | NaN | Documentary |
| **146141** | tt9916706 | Dankyavar Danka | 2013 | NaN | Comedy |
| **146142** | tt9916730 | 6 Gunn | 2017 | 116.0 | NaN |
| **146143** | tt9916754 | Chico Albuquerque - Revelações | 2013 | NaN | Documentary |

146144 rows × 5 columns

### 4.2.3 imdb_title_akas

In [367]:
```
#number of rows, columns and first 5 rows
display(imdb_title_akas.shape,imdb_title_akas.head())
```

(331703, 8)

| | title_id | ordering | title | region | language | types | attributes | is_original_title |
|---|---|---|---|---|---|---|---|---|
| **0** | tt0369610 | 10 | Джурасик свят | BG | bg | NaN | NaN | 0.0 |
| **1** | tt0369610 | 11 | Jurashikku warudo | JP | NaN | imdbDisplay | NaN | 0.0 |
| **2** | tt0369610 | 12 | Jurassic World: O Mundo dos Dinossauros | BR | NaN | imdbDisplay | NaN | 0.0 |
| **3** | tt0369610 | 13 | O Mundo dos Dinossauros | BR | NaN | NaN | short title | 0.0 |
| **4** | tt0369610 | 14 | Jurassic World | FR | NaN | imdbDisplay | NaN | 0.0 |

#### 4.2.3.1 Remove Columns

In [368]:
```
#remove unnecessary columns for future joins
imdb_title_akas = imdb_title_akas.drop(['ordering','language','typ
imdb_title_akas.head()
```

Out[368]:

| | title_id | title | is_original_title |
|---|---|---|---|
| **0** | tt0369610 | Джурасик свят | 0.0 |
| **1** | tt0369610 | Jurashikku warudo | 0.0 |
| **2** | tt0369610 | Jurassic World: O Mundo dos Dinossauros | 0.0 |
| **3** | tt0369610 | O Mundo dos Dinossauros | 0.0 |
| **4** | tt0369610 | Jurassic World | 0.0 |

#### 4.2.3.2 Check for Duplicates

In [369]:

```
#check for duplicates in title id
imdb_title_akas.loc[imdb_title_akas.duplicated(subset=['title_id']
```

Out[369]:

| | title_id | title | is_original_title |
|---|---|---|---|
| **0** | tt0369610 | Джурасик свят | 0.0 |
| **1** | tt0369610 | Jurashikku warudo | 0.0 |
| **2** | tt0369610 | Jurassic World: O Mundo dos Dinossauros | 0.0 |
| **3** | tt0369610 | O Mundo dos Dinossauros | 0.0 |
| **4** | tt0369610 | Jurassic World | 0.0 |
| **...** | ... | ... | ... |
| **331698** | tt9827784 | Sayonara kuchibiru | 1.0 |
| **331699** | tt9827784 | Farewell Song | 0.0 |
| **331700** | tt9880178 | La atención | 1.0 |
| **331701** | tt9880178 | La atención | 0.0 |
| **331702** | tt9880178 | The Attention | 0.0 |

254087 rows × 3 columns

In [370]:

```
imdb_title_akas.loc[imdb_title_akas['title_id'] == 'tt0369610']
```

Out[370]:

| | title_id | title | is_original_title |
|---|---|---|---|
| **0** | tt0369610 | Джурасик свят | 0.0 |
| **1** | tt0369610 | Jurashikku warudo | 0.0 |
| **2** | tt0369610 | Jurassic World: O Mundo dos Dinossauros | 0.0 |
| **3** | tt0369610 | O Mundo dos Dinossauros | 0.0 |
| **4** | tt0369610 | Jurassic World | 0.0 |
| **5** | tt0369610 | Jurassic World | 0.0 |
| **6** | tt0369610 | Jurassic World | 0.0 |
| **7** | tt0369610 | Jurski svijet | 0.0 |
| **8** | tt0369610 | Olam ha'Yura | 0.0 |
| **9** | tt0369610 | Jurassic World: Mundo Jurásico | 0.0 |
| **10** | tt0369610 | Jurassic World: Sauruste maailm | 0.0 |

Many duplicates found. Removing all rows where the is_original_title is 0.0

In [371]:
```
#only keep rows where is_original_title equal 1
imdb_title_akas = imdb_title_akas.loc[imdb_title_akas['is_original
imdb_title_akas
```

Out[371]:

| | title_id | title | is_original_title |
|---|---|---|---|
| **38** | tt0369610 | Jurassic World | 1.0 |
| **80** | tt0401729 | John Carter | 1.0 |
| **83** | tt10010134 | Versailles Rediscovered - The Sun King's Vanis... | 1.0 |
| **86** | tt10027708 | Miguelito - Canto a Borinquen | 1.0 |
| **90** | tt10050722 | Thing I Don't Get | 1.0 |
| **...** | ... | ... | ... |
| **331690** | tt9723084 | Anderswo. Allein in Afrika | 1.0 |
| **331692** | tt9726638 | Monkey King: The Volcano | 1.0 |
| **331696** | tt9755806 | Big Shark | 1.0 |
| **331698** | tt9827784 | Sayonara kuchibiru | 1.0 |
| **331700** | tt9880178 | La atención | 1.0 |

44700 rows × 3 columns

In [372]:
```
#recheck for duplicates
imdb_title_akas.loc[imdb_title_akas.duplicated(subset=['title_id']
```

Out[372]:

| | title_id | title | is_original_title |
|---|---|---|---|
| **19255** | tt1226736 | Against the Wind | 1.0 |
| **19256** | tt1226736 | Alexander Jamieson | 1.0 |
| **23989** | tt2392386 | The Sugar Wars: The Life Story of Angelo Lonardo | 1.0 |
| **23990** | tt2392386 | Sugar Wars - The Rise of the Cleveland Mafia | 1.0 |
| **33369** | tt1754830 | Being Us | 1.0 |
| **33372** | tt1754830 | Us | 1.0 |
| **37514** | tt2445698 | Entre Nós | 1.0 |
| **37517** | tt2445698 | A Pele do Cordeiro | 1.0 |
| **42571** | tt2219210 | Crawl Bitch Crawl | 1.0 |
| **42574** | tt2219210 | Crawl or Die | 1.0 |
| **63392** | tt1842446 | Rafina | 1.0 |

In [373]:
```
#example check
imdb_title_akas.loc[imdb_title_akas['title_id'] == 'tt2219210']
```

Out[373]:

| | title_id | title | is_original_title |
|---|---|---|---|
| **42571** | tt2219210 | Crawl Bitch Crawl | 1.0 |
| **42574** | tt2219210 | Crawl or Die | 1.0 |

Still have a small number of duplicates. Will remove these rows now

In [374]:
```
#remove duplicates for rows with duplicate title_id
imdb_title_akas.drop_duplicates(subset=['title_id'], inplace=True)
imdb_title_akas
```

Out[374]:

| | title_id | title | is_original_title |
|---|---|---|---|
| **38** | tt0369610 | Jurassic World | 1.0 |
| **80** | tt0401729 | John Carter | 1.0 |
| **83** | tt10010134 | Versailles Rediscovered - The Sun King's Vanis... | 1.0 |
| **86** | tt10027708 | Miguelito - Canto a Borinquen | 1.0 |
| **90** | tt10050722 | Thing I Don't Get | 1.0 |
| **...** | ... | ... | ... |
| **331690** | tt9723084 | Anderswo. Allein in Afrika | 1.0 |
| **331692** | tt9726638 | Monkey King: The Volcano | 1.0 |
| **331696** | tt9755806 | Big Shark | 1.0 |
| **331698** | tt9827784 | Sayonara kuchibiru | 1.0 |
| **331700** | tt9880178 | La atención | 1.0 |

44653 rows × 3 columns

In [375]:
```
#recheck for duplicates
imdb_title_akas.loc[imdb_title_akas.duplicated(subset=['title_id']
```

Out[375]:

| title_id | title | is_original_title |
|---|---|---|

No more duplicate title_id's

#### 4.2.3.3 Final Table View

In [376]:
```
imdb_title_akas
```

Out[376]:

| | title_id | title | is_original_title |
|---|---|---|---|
| **38** | tt0369610 | Jurassic World | 1.0 |
| **80** | tt0401729 | John Carter | 1.0 |
| **83** | tt10010134 | Versailles Rediscovered - The Sun King's Vanis... | 1.0 |
| **86** | tt10027708 | Miguelito - Canto a Borinquen | 1.0 |
| **90** | tt10050722 | Thing I Don't Get | 1.0 |
| **...** | ... | ... | ... |
| **331690** | tt9723084 | Anderswo. Allein in Afrika | 1.0 |
| **331692** | tt9726638 | Monkey King: The Volcano | 1.0 |
| **331696** | tt9755806 | Big Shark | 1.0 |
| **331698** | tt9827784 | Sayonara kuchibiru | 1.0 |
| **331700** | tt9880178 | La atención | 1.0 |

44653 rows × 3 columns

# 4.3 Joining Tables

I will be joining the tables in the following order:

- imdb_title_basics
- imdb_title_akas
- tn_movie_budgets
- imdb_title_ratings
- imdb_title_principals
- imdb_title_crew
- imdb_name_basics (TBD)

## 4.3.1 Join imdb_title_basics and imdb_title_akas

I am starting with this join because there are many duplicate primary_titles in the imdb_title_basics table. I will use the is_original_title field to filter down the titles before joining to imdb_title_basics. This will ensure that when I join with tn_movie_budgets that I am not applying financials to the wrong movies with identical names.

I will be joining these tables on:

- imdb_title_basics: tconst
- imdb_title_akas: title_id

In [377]:
```
#review the shape of the dataframes prior to join
display(imdb_title_basics.shape, imdb_title_akas.shape)
```

```
(146144, 5)
```

```
(44653, 3)
```

#### 4.3.1.1 Join the tables

In [378]:
```
movies_df = imdb_title_basics.merge(imdb_title_akas,how='inner',le
movies_df
```

Out[378]:

| | tconst | primary_title | start_year | runtime_minutes | genres | titl |
|---|---|---|---|---|---|---|
| **0** | tt0063540 | Sunghursh | 2013 | 175.0 | Action,Crime,Drama | tt0063 |
| **1** | tt0066787 | One Day Before the Rainy Season | 2019 | 114.0 | Biography,Drama | tt0066 |
| **2** | tt0069049 | The Other Side of the Wind | 2018 | 122.0 | Drama | tt0069 |
| **3** | tt0069204 | Sabse Bada Sukh | 2018 | NaN | Comedy,Drama | tt0069 |
| **4** | tt0100275 | The Wandering Soap Opera | 2017 | 80.0 | Comedy,Drama,Fantasy | tt0100 |
| **...** | ... | ... | ... | ... | ... | |
| **44648** | tt9911774 | Padmavyuhathile Abhimanyu | 2019 | 130.0 | Drama | tt9911 |

#### 4.3.1.2 Check for Duplicates

In [379]:
```
#check for duplicates in the primary_title field
movies_df.loc[movies_df.duplicated(subset=(['tconst']))]
```

Out[379]:

| tconst | primary_title | start_year | runtime_minutes | genres | title_id | title | is_original_title |
|---|---|---|---|---|---|---|---|

#### 4.3.1.3 Remove Duplicates

In [380]:
```
#delete some of the duplicates after manually reviewing
movies_df.drop([32,1226,1500,1501,2238,3340,5070,6278,6877,7433,75
                12026,12792,14022,14049,14252,14402,14788,15646,16
               17003,17384,17890,17905,18396,18611,19072,19073,198
               21259,21870,22843,24549,26617,26634,27813,31341,315
               35258,37100,38852,41980,43111], inplace=True)
```

In [381]:
```
#recheck for duplicates
movies_df.loc[movies_df.duplicated(subset=(['tconst']))]
```

Out[381]:

| tconst | primary_title | start_year | runtime_minutes | genres | title_id | title | is_original_title |
|---|---|---|---|---|---|---|---|

No more duplicates

#### 4.3.1.4 Final Table View

In [382]:
```
movies_df
```

Out[382]:

| | tconst | primary_title | start_year | runtime_minutes | genres | title_i |
|---|---|---|---|---|---|---|
| **0** | tt0063540 | Sunghursh | 2013 | 175.0 | Action,Crime,Drama | tt006354 |
| **1** | tt0066787 | One Day Before the Rainy Season | 2019 | 114.0 | Biography,Drama | tt006678 |
| **2** | tt0069049 | The Other Side of the Wind | 2018 | 122.0 | Drama | tt006904 |
| **3** | tt0069204 | Sabse Bada Sukh | 2018 | NaN | Comedy,Drama | tt006920 |
| **4** | tt0100275 | The Wandering Soap Opera | 2017 | 80.0 | Comedy,Drama,Fantasy | tt010027 |
| **...** | ... | ... | ... | ... | ... | . |
| **44648** | tt9911774 | Padmavyuhathile Abhimanyu | 2019 | 130.0 | Drama | tt991177 |
| **44649** | tt9913248 | Nepal - Homebird | 2019 | 52.0 | Documentary | tt991324 |
| **44650** | tt9914254 | A Cherry Tale | 2019 | 85.0 | Documentary | tt991425 |
| **44651** | tt9915436 | Vida em Movimento | 2019 | 70.0 | Documentary | tt991543 |
| **44652** | tt9916170 | The Rehearsal | 2019 | 51.0 | Drama | tt991617 |

44606 rows × 8 columns

## 4.3.2 Join movies_df and tn_movie_budgets

I will be joining these tables on:

- movies_df: [primary_title, start_year]
- tn_movie_budgets: [movie, start_year]

In [383]:
```
#review the shape of the dataframes prior to join
display(movies_df.shape, tn_movie_budgets.shape)
```

```
(44606, 8)

(5782, 6)
```

#### 4.3.2.1 Join The Tables

In [384]:

```python
#join tables together
movies_df = movies_df.merge(tn_movie_budgets, how='inner',
                    left_on=['primary_title','start_year'],
                    right_on=['movie','start_year'])
movies_df
```

Out[384]:

| | tconst | primary_title | start_year | runtime_minutes | genres | title_i |
|---|---|---|---|---|---|---|
| **0** | tt0249516 | Foodfight! | 2012 | 91.0 | Action,Animation,Comedy | tt024951 |
| **1** | tt0359950 | The Secret Life of Walter Mitty | 2013 | 114.0 | Adventure,Comedy,Drama | tt035995 |
| **2** | tt0365907 | A Walk Among the Tombstones | 2014 | 114.0 | Action,Crime,Drama | tt036590 |
| **3** | tt0369610 | Jurassic World | 2015 | 124.0 | Action,Adventure,Sci-Fi | tt036961 |
| **4** | tt0376136 | The Rum Diary | 2011 | 119.0 | Comedy,Drama | tt037613 |
| **...** | ... | ... | ... | ... | ... | . |
| **1444** | tt8155288 | Happy Death | 2019 | 100.0 | Drama,Horror,Mystery | tt815528 |

#### 4.3.2.2 Check for Duplicates

In [385]:
```
#check for duplicates
movies_df.loc[movies_df.duplicated(subset=['primary_title', 'start
                                    keep=False)]
```

Out[385]:

| | tconst | primary_title | start_year | runtime_minutes | genres | titl |
|---|---|---|---|---|---|---|
| 156 | tt1085492 | The Prince | 2014 | 93.0 | Action,Thriller | tt108 |
| 157 | tt3918106 | The Prince | 2014 | 71.0 | NaN | tt391 |
| 158 | tt4161288 | The Prince | 2014 | 92.0 | Drama | tt416 |
| 220 | tt1216492 | Leap Year | 2010 | 100.0 | Comedy,Romance | tt121 |
| 221 | tt1537401 | Leap Year | 2010 | 94.0 | Drama,Romance | tt153 |
| 313 | tt1327709 | Cyrus | 2010 | 87.0 | Crime,Horror,Mystery | tt132 |
| 314 | tt1336617 | Cyrus | 2010 | 91.0 | Comedy,Drama,Romance | tt133 |
| 474 | tt1554091 | A Better Life | 2011 | 98.0 | Drama,Romance | tt155 |
| 475 | tt2027265 | A Better Life | 2011 | 110.0 | Drama | tt202 |

In [386]:
```
#example check
movies_df.loc[movies_df['primary_title'] == 'The Prince']
```

Out[386]:

| | tconst | primary_title | start_year | runtime_minutes | genres | title_id | title | is_o |
|---|---|---|---|---|---|---|---|---|
| 156 | tt1085492 | The Prince | 2014 | 93.0 | Action,Thriller | tt1085492 | The Prince | |
| 157 | tt3918106 | The Prince | 2014 | 71.0 | NaN | tt3918106 | Ksiaze | |
| 158 | tt4161288 | The Prince | 2014 | 92.0 | Drama | tt4161288 | Shah-zadeh | |

#### 4.3.2.3 Remove Duplicates

In [387]:
```
#remove 39 duplicates
movies_df.drop_duplicates(subset=['primary_title','start_year'],ir
```

In [388]:
```
#recheck the duplicates
movies_df.loc[movies_df.duplicated(subset=['primary_title', 'start
                                  keep=False)]
```

Out[388]:

| | tconst | primary_title | start_year | runtime_minutes | genres | title_id | title | is_original_title | relea |
|---|---|---|---|---|---|---|---|---|---|

No duplicates for primary_title and start_year as well as tconst

#### 4.3.2.4 Final Table View

In [389]:
```
movies_df
```

Out[389]:

| | tconst | primary_title | start_year | runtime_minutes | genres | title_id |
|---|---|---|---|---|---|---|
| **0** | tt0249516 | Foodfight! | 2012 | 91.0 | Action,Animation,Comedy | tt0249516 |
| **1** | tt0359950 | The Secret Life of Walter Mitty | 2013 | 114.0 | Adventure,Comedy,Drama | tt0359950 |
| **2** | tt0365907 | A Walk Among the Tombstones | 2014 | 114.0 | Action,Crime,Drama | tt0365907 |
| **3** | tt0369610 | Jurassic World | 2015 | 124.0 | Action,Adventure,Sci-Fi | tt0369610 |
| **4** | tt0376136 | The Rum Diary | 2011 | 119.0 | Comedy,Drama | tt0376136 |
| **...** | ... | ... | ... | ... | ... | ... |
| **1444** | tt8155288 | Happy Death Day 2U | 2019 | 100.0 | Drama,Horror,Mystery | tt8155288 |
| **1445** | tt8266310 | Blinded by the Light | 2019 | 117.0 | Biography,Comedy,Drama | tt8266310 |
| **1446** | tt8364368 | Crawl | 2019 | NaN | Action,Horror,Thriller | tt8364368 |
| **1447** | tt8632862 | Fahrenheit 11/9 | 2018 | 128.0 | Documentary | tt8632862 |
| **1448** | tt9024106 | Unplanned | 2019 | 106.0 | Biography,Drama | tt9024106 |

1428 rows × 13 columns

### 4.3.3 Join movies_df and imdb_title_ratings

I will be joining these tables on:

- movies_df: [tconst]
- imdb_title_ratings: [tconst]

In [390]:
```
#review the shape of the dataframes prior to join
display(movies_df.shape, imdb_title_ratings.shape)
```

```
(1428, 13)

(73856, 3)
```

#### 4.3.3.1 Join the Tables

In [391]:

```
#join tables together
movies_df = movies_df.merge(imdb_title_ratings, how='inner', on='t
movies_df
```

Out[391]:

| | tconst | primary_title | start_year | runtime_minutes | genres | title_id |
|---|---|---|---|---|---|---|
| **0** | tt0249516 | Foodfight! | 2012 | 91.0 | Action,Animation,Comedy | tt0249516 |
| **1** | tt0359950 | The Secret Life of Walter Mitty | 2013 | 114.0 | Adventure,Comedy,Drama | tt0359950 |
| **2** | tt0365907 | A Walk Among the Tombstones | 2014 | 114.0 | Action,Crime,Drama | tt0365907 |
| **3** | tt0369610 | Jurassic World | 2015 | 124.0 | Action,Adventure,Sci-Fi | tt0369610 |
| **4** | tt0376136 | The Rum Diary | 2011 | 119.0 | Comedy,Drama | tt0376136 |
| **...** | ... | ... | ... | ... | ... | ... |
| **1413** | tt8043306 | Teefa in Trouble | 2018 | 155.0 | Action,Comedy,Crime | tt8043306 |
| **1414** | tt8155288 | Happy Death Day 2U | 2019 | 100.0 | Drama,Horror,Mystery | tt8155288 |
| **1415** | tt8266310 | Blinded by the Light | 2019 | 117.0 | Biography,Comedy,Drama | tt8266310 |
| **1416** | tt8632862 | Fahrenheit 11/9 | 2018 | 128.0 | Documentary | tt8632862 |
| **1417** | tt9024106 | Unplanned | 2019 | 106.0 | Biography,Drama | tt9024106 |

1418 rows × 15 columns

#### 4.3.3.2 Final Table View

In [392]:
```
movies_df
```

Out[392]:

| | tconst | primary_title | start_year | runtime_minutes | genres | title_id |
|---|---|---|---|---|---|---|
| **0** | tt0249516 | Foodfight! | 2012 | 91.0 | Action,Animation,Comedy | tt0249516 |
| **1** | tt0359950 | The Secret Life of Walter Mitty | 2013 | 114.0 | Adventure,Comedy,Drama | tt0359950 |
| **2** | tt0365907 | A Walk Among the Tombstones | 2014 | 114.0 | Action,Crime,Drama | tt0365907 |
| **3** | tt0369610 | Jurassic World | 2015 | 124.0 | Action,Adventure,Sci-Fi | tt0369610 |
| **4** | tt0376136 | The Rum Diary | 2011 | 119.0 | Comedy,Drama | tt0376136 |
| **...** | ... | ... | ... | ... | ... | ... |
| **1413** | tt8043306 | Teefa in Trouble | 2018 | 155.0 | Action,Comedy,Crime | tt8043306 |
| **1414** | tt8155288 | Happy Death Day 2U | 2019 | 100.0 | Drama,Horror,Mystery | tt8155288 |
| **1415** | tt8266310 | Blinded by the Light | 2019 | 117.0 | Biography,Comedy,Drama | tt8266310 |
| **1416** | tt8632862 | Fahrenheit 11/9 | 2018 | 128.0 | Documentary | tt8632862 |
| **1417** | tt9024106 | Unplanned | 2019 | 106.0 | Biography,Drama | tt9024106 |

1418 rows × 15 columns

## 4.3.4 Join movies_df and bom_movies_gross

I will be joining these tables on:

- movies_df: [movie,start_year]
- bom_movies_gross: [title,year]

In [393]:

```
#join tables
movies_df = movies_df.merge(bom_movie_gross, how='inner', left_on=
movies_df
```

Out[393]:

| | tconst | primary_title | start_year | runtime_minutes | genres | title_id |
|---|---|---|---|---|---|---|
| **0** | tt0359950 | The Secret Life of Walter Mitty | 2013 | 114.0 | Adventure,Comedy,Drama | tt0359950 |
| **1** | tt0365907 | A Walk Among the Tombstones | 2014 | 114.0 | Action,Crime,Drama | tt0365907 |
| **2** | tt0369610 | Jurassic World | 2015 | 124.0 | Action,Adventure,Sci-Fi | tt0369610 |
| **3** | tt0376136 | The Rum Diary | 2011 | 119.0 | Comedy,Drama | tt0376136 |
| **4** | tt0383010 | The Three Stooges | 2012 | 92.0 | Comedy,Family | tt0383010 |
| **...** | ... | ... | ... | ... | ... | ... |
| **1018** | tt7388562 | Paul, Apostle of Christ | 2018 | 108.0 | Adventure,Biography,Drama | tt7388562 |
| **1019** | tt7401588 | Instant Family | 2018 | 118.0 | Comedy,Drama | tt7401588 |
| **1020** | tt7535780 | The Great Wall | 2017 | 72.0 | Documentary | tt7535780 |
| **1021** | tt7784604 | Hereditary | 2018 | 127.0 | Drama,Horror,Mystery | tt7784604 |
| **1022** | tt7959026 | The Mule | 2018 | 116.0 | Crime,Drama,Thriller | tt7959026 |

1023 rows × 20 columns

### 4.3.5 Join movies_df and imdb_title_principals (not proceeding yet)

I will be joining these tables on:

- movies_df: [tconst]
- imdb_title_principals: [tconst]

In [394]:
```
#review the shape of the dataframes prior to join
display(movies_df.shape, imdb_title_principals.shape)
```

```
(1023, 20)

(1028186, 6)
```

**Question: How do I deal with the expansion of rows when trying to bring in directors and writers?**

# 4.4 Post-Join Clean-Up of movies_df

It is now time to clean up the joined dataset in order to minimize noise in the data. This will include looking for:

- deleting columns
- deleting rows
- duplicates
- nulls
- changing data types

In [395]:

```
movies_df
```

Out[395]:

| | tconst | primary_title | start_year | runtime_minutes | genres | title_id |
|---|---|---|---|---|---|---|
| **0** | tt0359950 | The Secret Life of Walter Mitty | 2013 | 114.0 | Adventure,Comedy,Drama | tt0359950 |
| **1** | tt0365907 | A Walk Among the Tombstones | 2014 | 114.0 | Action,Crime,Drama | tt0365907 |
| **2** | tt0369610 | Jurassic World | 2015 | 124.0 | Action,Adventure,Sci-Fi | tt0369610 |
| **3** | tt0376136 | The Rum Diary | 2011 | 119.0 | Comedy,Drama | tt0376136 |
| **4** | tt0383010 | The Three Stooges | 2012 | 92.0 | Comedy,Family | tt0383010 |
| **...** | ... | ... | ... | ... | ... | ... |
| **1018** | tt7388562 | Paul, Apostle of Christ | 2018 | 108.0 | Adventure,Biography,Drama | tt7388562 |
| **1019** | tt7401588 | Instant Family | 2018 | 118.0 | Comedy,Drama | tt7401588 |
| **1020** | tt7535780 | The Great Wall | 2017 | 72.0 | Documentary | tt7535780 |
| **1021** | tt7784604 | Hereditary | 2018 | 127.0 | Drama,Horror,Mystery | tt7784604 |
| **1022** | tt7959026 | The Mule | 2018 | 116.0 | Crime,Drama,Thriller | tt7959026 |

1023 rows × 20 columns

### 4.4.1 Deleting Rows

I want to look at the statistics around production budget to ensure that the dataset is not dominated by movies from indie studios. In order to determine which studios are considered the "top studios", I will join the tn_movie_budgets and bom_movie_gross tables and find the studios which have had at least one film in the top 50% of production budget.

I will be joining these tables on:

- tn_movie_budgets: [movie,start_year]
- bom_movie_gross: [title,year]

In [396]:

```
#join the tables
prod_budget_df = tn_movie_budgets.merge(bom_movie_gross, how='inne
                                        left_on=['movie','start_y
                                        right_on=['title','year']

prod_budget_df
```

Out[396]:

| | release_date | movie | production_budget | domestic_gross_x | worldwide_gross | start_ye |
|---|---|---|---|---|---|---|
| **0** | 2011-05-20 | Pirates of the Caribbean: On Stranger Tides | 410600000 | 241063875 | 1045663875 | 20 |
| **1** | 2015-05-01 | Avengers: Age of Ultron | 330600000 | 459005868 | 1403013963 | 20 |
| **2** | 2018-04-27 | Avengers: Infinity War | 300000000 | 678815482 | 2048134200 | 20 |
| **3** | 2017-11-17 | Justice League | 300000000 | 229024295 | 655945209 | 20 |
| **4** | 2015-11-06 | Spectre | 300000000 | 200074175 | 879620923 | 20 |
| **...** | ... | ... | ... | ... | ... | |
| **1210** | 2012-04-27 | Sound of My Voice | 135000 | 408015 | 429448 | 20 |
| **1211** | 2012-06-15 | Your Sister's Sister | 120000 | 1597486 | 3090593 | 20 |
| **1212** | 2015-07-10 | The Gallows | 100000 | 22764410 | 41656474 | 20 |
| **1213** | 2017-07-07 | A Ghost Story | 100000 | 1594798 | 2769782 | 20 |
| **1214** | 2010-11-12 | Tiny Furniture | 50000 | 391674 | 424149 | 20 |

1215 rows × 11 columns

In [397]:
```
#what are the studios in the top 50% of spend
top_studio_list = set(list(prod_budget_df.loc[prod_budget_df['proc
                                              > prod_budget_df['pr
                                              quantile(q=0.5)]['st
```

In [398]:
```
#number and names of unique top studios
print(len(top_studio_list))
list(top_studio_list)
```

```
40
```

Out[398]:
```
['MNE',
 'Sony',
 'STX',
 'SPC',
 'Annapurna',
 'WB',
 'Magn.',
 'CBS',
 'MGM',
 'BG',
 'EOne',
 'Focus',
 'TriS',
 'RAtt.',
 'Fox',
 'Gold.',
 'SGem',
 'BST',
 'Uni.',
 'WB (NL)',
 'GrtIndia',
 'Free',
 'Wein.',
 'Studio 8',
 'RTWC',
 'Par.',
 'BV',
 'LG/S',
 'Sum.',
 'NM',
 'P/DW',
 'W/Dim.',
 'BSC',
 'Rela.',
 'FoxS',
 'ENTMP',
 'LGF',
 'FD',
 'VE',
 'ORF']
```

In [399]:
```
#filter movies_df to include only movies from top_studios_list
movies_df = movies_df.loc[movies_df['studio'].isin(top_studio_list
movies_df
```

Out[399]:

| | tconst | primary_title | start_year | runtime_minutes | genres | title_ |
|---|---|---|---|---|---|---|
| **0** | tt0359950 | The Secret Life of Walter Mitty | 2013 | 114.0 | Adventure,Comedy,Drama | tt03599 |
| **1** | tt0365907 | A Walk Among the Tombstones | 2014 | 114.0 | Action,Crime,Drama | tt03659 |
| **2** | tt0369610 | Jurassic World | 2015 | 124.0 | Action,Adventure,Sci-Fi | tt03696 |
| **3** | tt0376136 | The Rum Diary | 2011 | 119.0 | Comedy,Drama | tt03761 |
| **4** | tt0383010 | The Three Stooges | 2012 | 92.0 | Comedy,Family | tt03830 |
| **...** | ... | ... | ... | ... | ... | |
| **1016** | tt7334528 | Uncle Drew | 2018 | 103.0 | Comedy,Sport | tt73345 |
| **1017** | tt7349662 | BlacKkKlansman | 2018 | 135.0 | Biography,Crime,Drama | tt73496 |
| **1019** | tt7401588 | Instant Family | 2018 | 118.0 | Comedy,Drama | tt74015 |
| **1020** | tt7535780 | The Great Wall | 2017 | 72.0 | Documentary | tt75357 |
| **1022** | tt7959026 | The Mule | 2018 | 116.0 | Crime,Drama,Thriller | tt79590 |

945 rows × 20 columns

## 4.4.2 Deleting Columns

In [400]:
```
#remove columns
movies_df = movies_df.drop(['title_x','is_original_title','title_y
movies_df
```

Out[400]:

| | tconst | start_year | runtime_minutes | genres | release_date | r |
|---|---|---|---|---|---|---|
| **0** | tt0359950 | 2013 | 114.0 | Adventure,Comedy,Drama | 2013-12-25 | The Secre of Walter |
| **1** | tt0365907 | 2014 | 114.0 | Action,Crime,Drama | 2014-09-19 | A Walk A the Tombs |
| **2** | tt0369610 | 2015 | 124.0 | Action,Adventure,Sci-Fi | 2015-06-12 | Jurassic ' |
| **3** | tt0376136 | 2011 | 119.0 | Comedy,Drama | 2011-10-28 | The Rum |
| **4** | tt0383010 | 2012 | 92.0 | Comedy,Family | 2012-04-13 | The ' Stc |
| **...** | ... | ... | ... | ... | ... | |
| **1016** | tt7334528 | 2018 | 103.0 | Comedy,Sport | 2018-06-29 | Uncle |
| **1017** | tt7349662 | 2018 | 135.0 | Biography,Crime,Drama | 2018-08-10 | BlacKkKlan |
| **1019** | tt7401588 | 2018 | 118.0 | Comedy,Drama | 2018-11-16 | Instant F |
| **1020** | tt7535780 | 2017 | 72.0 | Documentary | 2017-02-17 | The Grea |
| **1022** | tt7959026 | 2018 | 116.0 | Crime,Drama,Thriller | 2018-12-14 | The |

945 rows × 13 columns

In [401]:
```
#rename columns
movies_df = movies_df.rename(columns={'domestic_gross_x':'domestic
```

QUESTION:Do higher ratings correlate with more revenue?
ANSWER: there is a weak positive correlation between average rating and domestic_gross

### 4.4.3 Fixing Duplicates

In [402]:
```
#checking the number of duplicates in the dataframe
movies_df.duplicated().sum()
```

Out[402]: 0

### 4.4.4 Fixing Nulls

In [403]:
```
#checking the number of null values in each column
movies_df.isna().sum()
```

Out[403]:
```
tconst              0
start_year          0
runtime_minutes     0
genres              0
release_date        0
movie               0
production_budget   0
domestic_gross      0
worldwide_gross     0
averagerating       0
numvotes            0
studio              0
year                0
dtype: int64
```

In [404]:
```
#checking for zeros
movies_df.loc[(movies_df['domestic_gross'] == 0) & (movies_df['wor
```

Out[404]:
```
tconst              0
start_year          0
runtime_minutes     0
genres              0
release_date        0
movie               0
production_budget   0
domestic_gross      0
worldwide_gross     0
averagerating       0
numvotes            0
studio              0
year                0
dtype: int64
```

### 4.4.5 Changing Data Types

In [405]:
```
#checking the current data types of all columns
display(movies_df.info(),movies_df.head(2))
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 945 entries, 0 to 1022
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   tconst             945 non-null    object
 1   start_year         945 non-null    int64
 2   runtime_minutes    945 non-null    float64
 3   genres             945 non-null    object
 4   release_date       945 non-null    datetime64[ns]
 5   movie              945 non-null    object
 6   production_budget  945 non-null    int64
 7   domestic_gross     945 non-null    int64
 8   worldwide_gross    945 non-null    int64
 9   averagerating      945 non-null    float64
 10  numvotes           945 non-null    int64
 11  studio             945 non-null    object
 12  year               945 non-null    int64
dtypes: datetime64[ns](1), float64(2), int64(6), object(4)
memory usage: 103.4+ KB
```

None

| | tconst | start_year | runtime_minutes | genres | release_date | movie | p |
|---|---|---|---|---|---|---|---|
| **0** | tt0359950 | 2013 | 114.0 | Adventure,Comedy,Drama | 2013-12-25 | The Secret Life of Walter Mitty | |
| **1** | tt0365907 | 2014 | 114.0 | Action,Crime,Drama | 2014-09-19 | A Walk Among the Tombstones | |

Data types are reasonable for the values

## 4.4.6 Adding Release Month

Adding a column for the release month will allow me to see which months are most common to release a movie

In [406]:
```
movies_df['release_month'] = movies_df['release_date'].dt.month
```

### 4.4.7 Final Table View

In [407]:
```
movies_df
```

Out[407]:

| | tconst | start_year | runtime_minutes | genres | release_date | r |
|---|---|---|---|---|---|---|
| **0** | tt0359950 | 2013 | 114.0 | Adventure,Comedy,Drama | 2013-12-25 | The Secre of Walter |
| **1** | tt0365907 | 2014 | 114.0 | Action,Crime,Drama | 2014-09-19 | A Walk A the Tombs |
| **2** | tt0369610 | 2015 | 124.0 | Action,Adventure,Sci-Fi | 2015-06-12 | Jurassic \ |
| **3** | tt0376136 | 2011 | 119.0 | Comedy,Drama | 2011-10-28 | The Rum |
| **4** | tt0383010 | 2012 | 92.0 | Comedy,Family | 2012-04-13 | The Stc |
| **...** | ... | ... | ... | ... | ... | |
| **1016** | tt7334528 | 2018 | 103.0 | Comedy,Sport | 2018-06-29 | Uncle |
| **1017** | tt7349662 | 2018 | 135.0 | Biography,Crime,Drama | 2018-08-10 | BlacKkKlan |
| **1019** | tt7401588 | 2018 | 118.0 | Comedy,Drama | 2018-11-16 | Instant F |
| **1020** | tt7535780 | 2017 | 72.0 | Documentary | 2017-02-17 | The Grea |
| **1022** | tt7959026 | 2018 | 116.0 | Crime,Drama,Thriller | 2018-12-14 | The |

945 rows × 14 columns

## 4.5 Calculations

### 4.5.1 Calculating Return on Investment

Return on Investment is the quantitative metric I am using to determine which movies are historically successful. This metric takes into account how much was invested to make the film and how much more revenue was received versus that cost. Also, return on investment does not need to be inflation adjusted.
Return on Investment takes the amount of profit (worldwide_gross - production_budget) and divides it by the initial investment cost (production_budget). This metric will give a sense of which movies were successful relative to how much they spent instead of making it an absolute metric on total profit. Later I will analyze if you can spending more is effective in profiting more.

To calculate Return on Investment, I will use the following equation:
(worldwide_gross - production_budget) / (production_budget)
Why am i using worldwide_gross vs domestic_gross?

In [408]:

```
#calculate roi and place into a new column
movies_df['roi'] = (movies_df['worldwide_gross']
                     - movies_df['production_budget']) / movies_df[
```

In [409]:

```
#check the new column
movies_df.sort_values('roi',ascending=False).head()
```

Out[409]:

| | tconst | start_year | runtime_minutes | genres | release_date | movie | pro |
|---|---|---|---|---|---|---|---|
| **694** | tt2309260 | 2015 | 81.0 | Horror,Mystery,Thriller | 2015-07-10 | The Gallows | |
| **384** | tt1560985 | 2012 | 83.0 | Horror | 2012-01-06 | The Devil Inside | |
| **372** | tt1536044 | 2010 | 91.0 | Horror | 2010-10-20 | Paranormal Activity 2 | |
| **960** | tt5052448 | 2017 | 104.0 | Horror,Mystery,Thriller | 2017-02-24 | Get Out | |
| **605** | tt1991245 | 2012 | 86.0 | Horror,Mystery,Thriller | 2012-05-25 | Chernobyl Diaries | |

### 4.5.2 Calculating Worldwide Profit

I want to have visibility on how profitability varies with movie variables. I will focus on worldwide profit as I believe Microsoft should release globally based on potential for more revenue.

In [410]:
```
#create new ww_profit column
movies_df['ww_profit'] = movies_df['worldwide_gross'] - movies_df[
movies_df
```

Out[410]:

| | tconst | start_year | runtime_minutes | genres | release_date | n |
|---|---|---|---|---|---|---|
| **0** | tt0359950 | 2013 | 114.0 | Adventure,Comedy,Drama | 2013-12-25 | The Secre of Walter |
| **1** | tt0365907 | 2014 | 114.0 | Action,Crime,Drama | 2014-09-19 | A Walk A the Tombs |
| **2** | tt0369610 | 2015 | 124.0 | Action,Adventure,Sci-Fi | 2015-06-12 | Jurassic ' |
| **3** | tt0376136 | 2011 | 119.0 | Comedy,Drama | 2011-10-28 | The Rum |
| **4** | tt0383010 | 2012 | 92.0 | Comedy,Family | 2012-04-13 | The Sto |
| **...** | ... | ... | ... | ... | ... | |
| **1016** | tt7334528 | 2018 | 103.0 | Comedy,Sport | 2018-06-29 | Uncle |
| **1017** | tt7349662 | 2018 | 135.0 | Biography,Crime,Drama | 2018-08-10 | BlacKkKlan |
| **1019** | tt7401588 | 2018 | 118.0 | Comedy,Drama | 2018-11-16 | Instant F |
| **1020** | tt7535780 | 2017 | 72.0 | Documentary | 2017-02-17 | The Grea |
| **1022** | tt7959026 | 2018 | 116.0 | Crime,Drama,Thriller | 2018-12-14 | The |

945 rows × 16 columns

# 5 EDA and Visualization

I am going to explore the data and create specific visualizations based on questions I'd like to ask.

In [411]:
```
#current state of the dataframe
movies_df
```

Out[411]:

| | tconst | start_year | runtime_minutes | genres | release_date | r |
|---|---|---|---|---|---|---|
| 0 | tt0359950 | 2013 | 114.0 | Adventure,Comedy,Drama | 2013-12-25 | The Secre of Walter |
| 1 | tt0365907 | 2014 | 114.0 | Action,Crime,Drama | 2014-09-19 | A Walk A the Tombs |
| 2 | tt0369610 | 2015 | 124.0 | Action,Adventure,Sci-Fi | 2015-06-12 | Jurassic ' |
| 3 | tt0376136 | 2011 | 119.0 | Comedy,Drama | 2011-10-28 | The Rum |
| 4 | tt0383010 | 2012 | 92.0 | Comedy,Family | 2012-04-13 | The Stc |
| ... | ... | ... | ... | ... | ... | |
| 1016 | tt7334528 | 2018 | 103.0 | Comedy,Sport | 2018-06-29 | Uncle |
| 1017 | tt7349662 | 2018 | 135.0 | Biography,Crime,Drama | 2018-08-10 | BlacKkKlan |
| 1019 | tt7401588 | 2018 | 118.0 | Comedy,Drama | 2018-11-16 | Instant F |
| 1020 | tt7535780 | 2017 | 72.0 | Documentary | 2017-02-17 | The Grea |
| 1022 | tt7959026 | 2018 | 116.0 | Crime,Drama,Thriller | 2018-12-14 | The |

945 rows × 16 columns

## 5.1 Q1: How has the movie industry grown over time?

I want to understand the overall landscape of the movie industry and how it has done both domestically and worldwide.

In [412]:
```
#create a dataframe specifically for this graph
financials_df = movies_df[['start_year','domestic_gross','worldwid
                           'production_budget']]
financials_by_year = financials_df.groupby(by='start_year')[['dome
                                                                 'worl
                                                                 'proc
financials_by_year.reset_index(inplace=True)
```

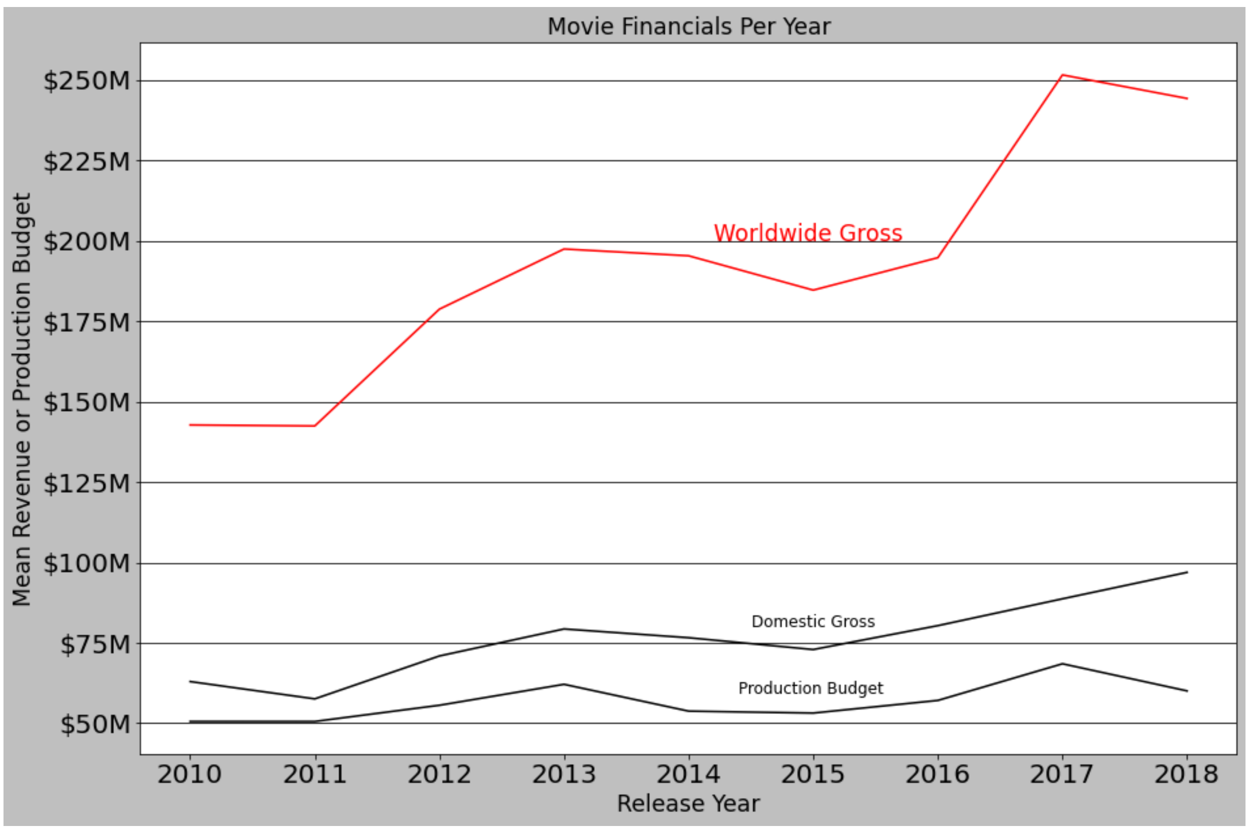
In [413]:

```
#view the dataframe
financials_by_year
```

Out[413]:

| | start_year | domestic_gross | worldwide_gross | production_budget |
|---|---|---|---|---|
| **0** | 2010 | 6.297399e+07 | 1.427252e+08 | 5.058240e+07 |
| **1** | 2011 | 5.758976e+07 | 1.424199e+08 | 5.057118e+07 |
| **2** | 2012 | 7.092585e+07 | 1.787259e+08 | 5.560105e+07 |
| **3** | 2013 | 7.932322e+07 | 1.974203e+08 | 6.211574e+07 |
| **4** | 2014 | 7.659738e+07 | 1.953114e+08 | 5.378010e+07 |
| **5** | 2015 | 7.292466e+07 | 1.846366e+08 | 5.317925e+07 |
| **6** | 2016 | 8.035124e+07 | 1.947212e+08 | 5.710727e+07 |
| **7** | 2017 | 8.869387e+07 | 2.515364e+08 | 6.849872e+07 |
| **8** | 2018 | 9.691103e+07 | 2.442186e+08 | 6.007000e+07 |

In [414]:
```
#create line graph of financials over time
fig, ax = plt.subplots(figsize=(15,10))
x = financials_by_year['start_year']
y1 = financials_by_year['domestic_gross']
y2 = financials_by_year['worldwide_gross']
y3 = financials_by_year['production_budget']

ax.plot(x,y1, label='Domestic Gross')
ax.plot(x,y2, label='Worldwide Gross', color='red')
ax.plot(x,y3, label='Production Budget', color='black')
ax.text(2014.4,59000000,'Production Budget', fontsize='large')
ax.text(2014.5,80000000,'Domestic Gross', fontsize='large')
ax.text(2014.2,200000000,'Worldwide Gross', color='red', fontsize=
ax.set_title('Movie Financials Per Year', fontsize='xx-large')
ax.set_xlabel('Release Year', fontsize='xx-large')
ax.set_ylabel('Mean Revenue or Production Budget', fontsize='xx-la
ax.yaxis.set_major_formatter(tick.FuncFormatter(reformat_large_tic
ax.grid(axis='y')
ax.tick_params(axis='both', which='major', labelsize=20)
```




**Analysis**

The graph shows that worldwide gross revenue has increased at a faster rate than domestic in the past 10 years. This informs us that a worldwide release is preferred over domestic only.

### 5.1.1 Calculate percentage increase in worldwide revenue in last 10 years

In [415]:
```
#calculating the percent difference between year 2018 and 2010 for
((financials_by_year['worldwide_gross'][8] -
financials_by_year['worldwide_gross'][0])/financials_by_year['worl
```

Out[415]: 71.11104004494872

**Analysis**
There has been a 71% increase in worldwide gross revenue from 2010-2018

### 5.1.2 Calculate percentage of movies that do not make their money back

In [416]:
```
#Calculate percentage of movies that do not make money
movies_df.loc[movies_df['roi'] <= 0]['roi'].count()/len(movies_df)
```

Out[416]: 14.920634920634921

**Analysis**
15% of movies do not make their money back

## 5.2 Q2: Which genres produced the best ROI?

I want understand which genres produce the best ROI historically. I will use median because of the presence of outliers which I believe should be kept in the dataset because they are accurate.

**Adding Genre Columns**

Splitting the genres into columns will allow for the analysis of financial information by genre to see which genres are most successful.

In [417]:

```
#df.explode can be used to create multiple rows
movies_df_genres = movies_df
movies_df_genres['genres'] = movies_df_genres['genres'].map(lambda
                                                              x.spli

movies_df_genres = movies_df.explode('genres')
movies_df_genres
```

Out[417]:

| | tconst | start_year | runtime_minutes | genres | release_date | movie | productio |
|---|---|---|---|---|---|---|---|
| **0** | tt0359950 | 2013 | 114.0 | Adventure | 2013-12-25 | The Secret Life of Walter Mitty | |
| **0** | tt0359950 | 2013 | 114.0 | Comedy | 2013-12-25 | The Secret Life of Walter Mitty | |
| **0** | tt0359950 | 2013 | 114.0 | Drama | 2013-12-25 | The Secret Life of Walter Mitty | |
| **1** | tt0365907 | 2014 | 114.0 | Action | 2014-09-19 | A Walk Among the Tombstones | |
| **1** | tt0365907 | 2014 | 114.0 | Crime | 2014-09-19 | A Walk Among the Tombstones | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **1019** | tt7401588 | 2018 | 118.0 | Drama | 2018-11-16 | Instant Family | |
| **1020** | tt7535780 | 2017 | 72.0 | Documentary | 2017-02-17 | The Great Wall | 1 |
| **1022** | tt7959026 | 2018 | 116.0 | Crime | 2018-12-14 | The Mule | |
| **1022** | tt7959026 | 2018 | 116.0 | Drama | 2018-12-14 | The Mule | |
| **1022** | tt7959026 | 2018 | 116.0 | Thriller | 2018-12-14 | The Mule | |

2497 rows × 16 columns

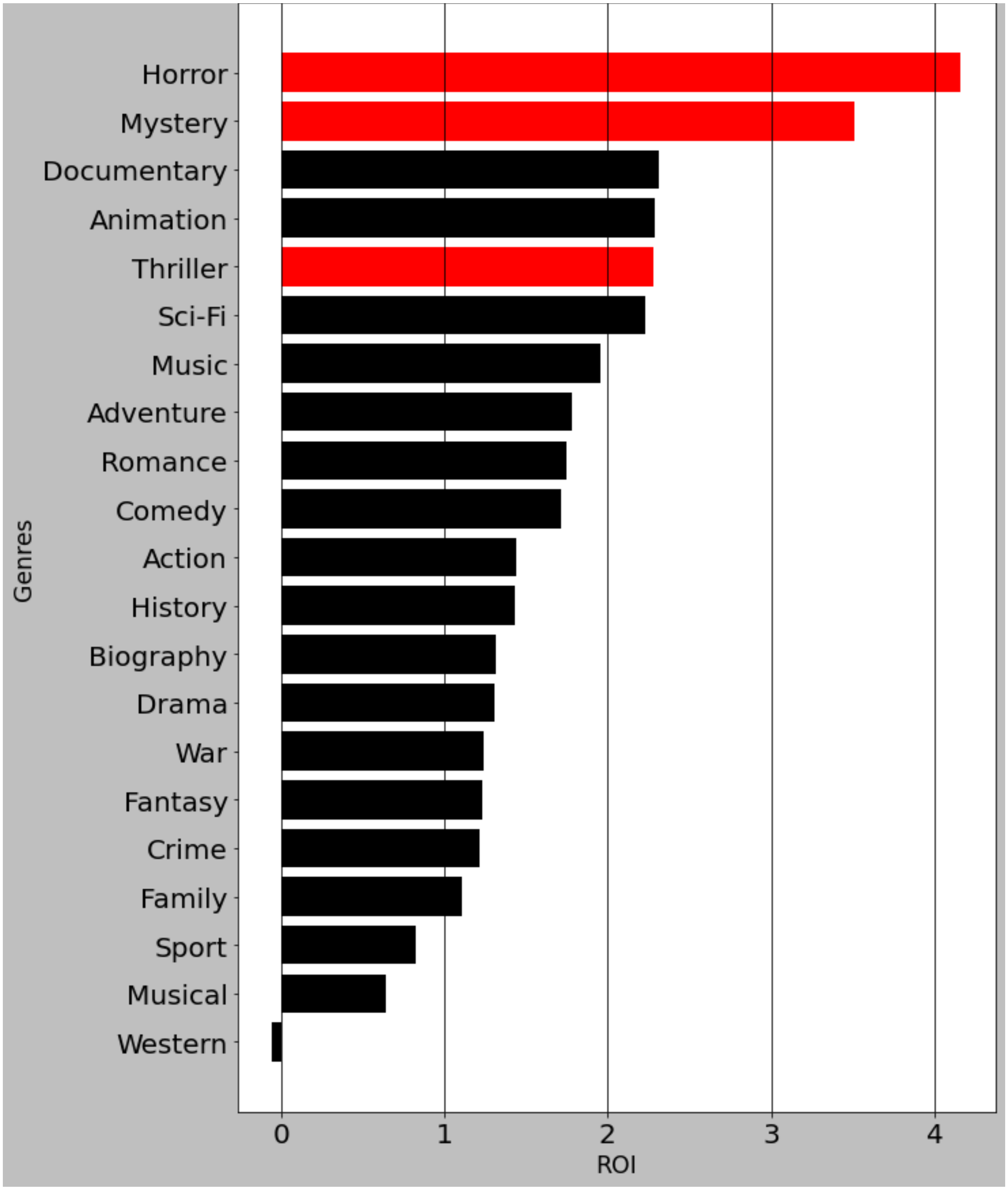
In [418]:
```
#create the necessary series of median roi by genre
roi_by_genres = movies_df_genres.groupby('genres').median()['roi']
roi_by_genres
```

Out[418]:
```
genres
Western       -0.054538
Musical        0.646412
Sport          0.822305
Family         1.106108
Crime          1.218164
Fantasy        1.234364
War            1.240222
Drama          1.306136
Biography      1.318634
History        1.431779
Action         1.443181
Comedy         1.711576
Romance        1.748406
Adventure      1.779868
Music          1.955618
Sci-Fi         2.231891
Thriller       2.278181
Animation      2.292976
Documentary    2.312333
Mystery        3.508959
Horror         4.163727
Name: roi, dtype: float64
```

In [419]:
```
#ordering roi by genre from most to least
genre_roi_order = movies_df_genres.groupby('genres').median()['roi
genre_roi_order_list = list(genre_roi_order)
genre_roi_order_list.reverse()
```

In [420]:
```
#create a barplot of median roi per genre
fig, ax = plt.subplots(figsize=(10,15))
y = genre_roi_order_list
width = roi_by_genres

ax.barh(y=y,width=width, color=['black','black','black','black','b
                                'black','black','black','black','b
                                'black','black','black','black','b
                                'black','black','red','red'])
ax.set_title('Median ROI by Genre',fontsize='xx-large')
ax.set_xlabel('ROI', fontsize='xx-large')
ax.set_ylabel('Genres', fontsize='xx-large')
ax.tick_params(axis='both', which='major', labelsize=20)
ax.grid(axis='x')
```

Median ROI by Genre




**Analysis**

The graph shows that Horror has twice the ROI over the median of other genres. This is mostly due to their low cost per film. I recommend that Microsoft start off safe and pick a primary genre of Horror and secondary genres of Mystery and Thriller to create a storyline.

### 5.2.1 Median Cost per Film Horror/Thriller/Mystery vs Others

I want to see what the median production cost is for a horror/mystery/thriller film vs other genres

In [421]:
```
#find the median cost per horror/thriller/mystery film
horror_production_cost = movies_df_genres.loc[movies_df_genres['ge
horror_production_cost.groupby('movie').median()['production_budge
```

Out[421]:
```
27250000.0
```

The median production cost of a Horror/Thriller/Mystery movie is $27,250,000

In [422]:
```
#find the median cost for all other types of genres
other_production_cost = movies_df_genres.loc[~movies_df_genres['ge
other_production_cost.groupby('movie').median()['production_budget
```

Out[422]:
```
35000000.0
```

In [423]:
```
horror_production_cost['genres'].value_counts()
```

Out[423]:
```
Thriller    163
Horror       89
Mystery      76
Name: genres, dtype: int64
```

The median production cost of a non-Horror/Thriller/Mystery movie is $35,000,000

**Analysis**

Horror/Thriller/Mystery movies are about 28% cheaper to produce

### 5.2.2 What if Microsoft wants to invest in the top 90% of production spend? What Genre?

In [424]:
```
movies_df
```

Out[424]:

| | tconst | start_year | runtime_minutes | genres | release_date | movie | proc |
|---|---|---|---|---|---|---|---|
| **0** | tt0359950 | 2013 | 114.0 | [Adventure, Comedy, Drama] | 2013-12-25 | The Secret Life of Walter Mitty | |
| **1** | tt0365907 | 2014 | 114.0 | [Action, Crime, Drama] | 2014-09-19 | A Walk Among the Tombstones | |
| **2** | tt0369610 | 2015 | 124.0 | [Action, Adventure, Sci-Fi] | 2015-06-12 | Jurassic World | |
| **3** | tt0376136 | 2011 | 119.0 | [Comedy, Drama] | 2011-10-28 | The Rum Diary | |
| **4** | tt0383010 | 2012 | 92.0 | [Comedy, Family] | 2012-04-13 | The Three Stooges | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **1016** | tt7334528 | 2018 | 103.0 | [Comedy, Sport] | 2018-06-29 | Uncle Drew | |
| **1017** | tt7349662 | 2018 | 135.0 | [Biography, Crime, Drama] | 2018-08-10 | BlacKkKlansman | |
| **1019** | tt7401588 | 2018 | 118.0 | [Comedy, Drama] | 2018-11-16 | Instant Family | |
| **1020** | tt7535780 | 2017 | 72.0 | [Documentary] | 2017-02-17 | The Great Wall | |
| **1022** | tt7959026 | 2018 | 116.0 | [Crime, Drama, Thriller] | 2018-12-14 | The Mule | |

945 rows × 16 columns

In [425]:
```
#look at the spend at the 90% percentile
movies_df_genres.groupby('movie').mean()['production_budget'].quar
```

Out[425]: 150000000.0

Top 90% of spend is $150,000,000

In [426]:
```
#filter movies_df_genres by a production spend of >= $150M
top_spend = movies_df_genres[movies_df_genres['production_budget']
```

In [427]:
```
#determine number of movies that fit production budget criteria by
top_spend.groupby('genres').count()['movie']
```

Out[427]:
```
genres
Action         76
Adventure      93
Animation      20
Comedy         19
Crime           2
Documentary     1
Drama          14
Family         11
Fantasy        23
History         2
Horror          4
Mystery         1
Sci-Fi         36
Thriller        6
Western         1
Name: movie, dtype: int64
```

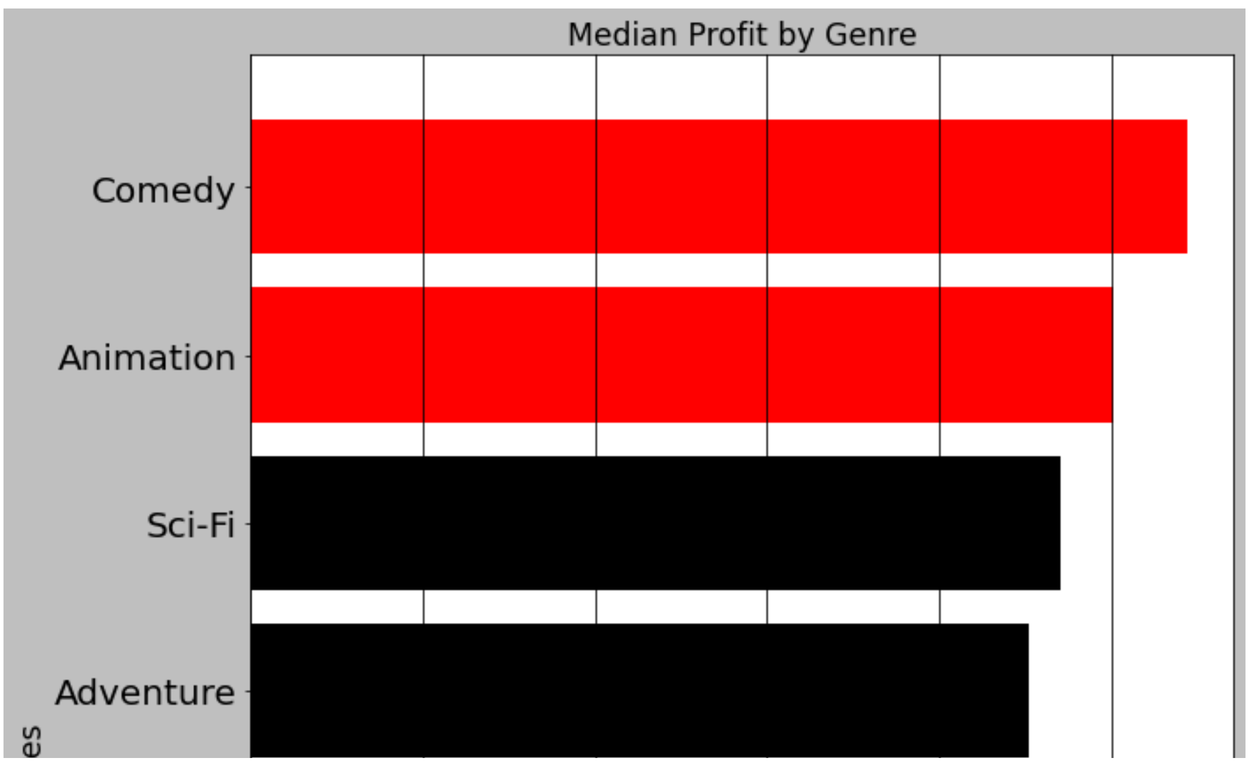
In [428]:
```
#going to eleminate any genres where the count is < 10
top_spend = top_spend.loc[~top_spend['genres'].isin(['Crime','Docu
                                              'History','Horror',
                                              'Thriller','Western
top_spend['genres'].value_counts()
```

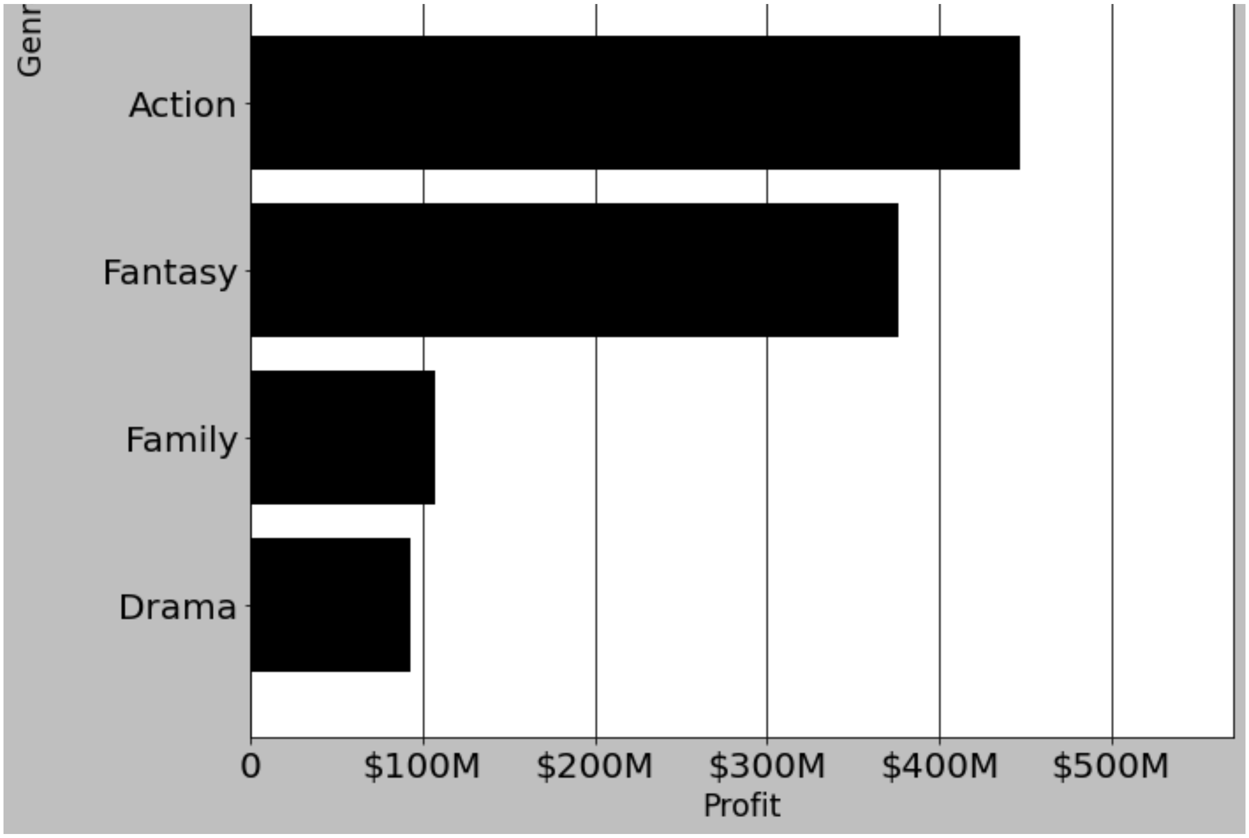
Out[428]:
```
Adventure    93
Action       76
Sci-Fi       36
Fantasy      23
Animation    20
Comedy       19
Drama        14
Family       11
Name: genres, dtype: int64
```

In [429]:
```
#create median profit dataframe
profit_top_spend = top_spend.groupby('genres').median()['ww_profit
profit_top_spend
```

Out[429]:
```
genres
Drama          93408207.0
Family        106928112.0
Fantasy       376072059.0
Action        447077953.5
Adventure     452220086.0
Sci-Fi        470071588.0
Animation     501177456.0
Comedy        543588329.0
Name: ww_profit, dtype: float64
```

In [430]:
```
#create a barplot of median roi per genre for high production cost
fig, ax = plt.subplots(figsize=(10,15))
y = profit_top_spend.index
width = profit_top_spend.sort_values()

ax.barh(y=y,width=width, color=['black','black','black','black','b
                                'black','red','red'])
ax.set_title('Median Profit by Genre',fontsize='xx-large')
ax.set_xlabel('Profit', fontsize='xx-large')
ax.set_ylabel('Genres', fontsize='xx-large')
ax.tick_params(axis='both', which='major', labelsize=20)
ax.grid(axis='x')
ax.xaxis.set_major_formatter(tick.FuncFormatter(reformat_large_tic
```






**Analysis**
The chart shows that Comedy and Animation movies return the best profit in movies which spend over $150M

In [431]:

```
#create roi series for graphing
roi_top_spend = top_spend.groupby('genres').median()['roi'].sort_v
roi_top_spend
```

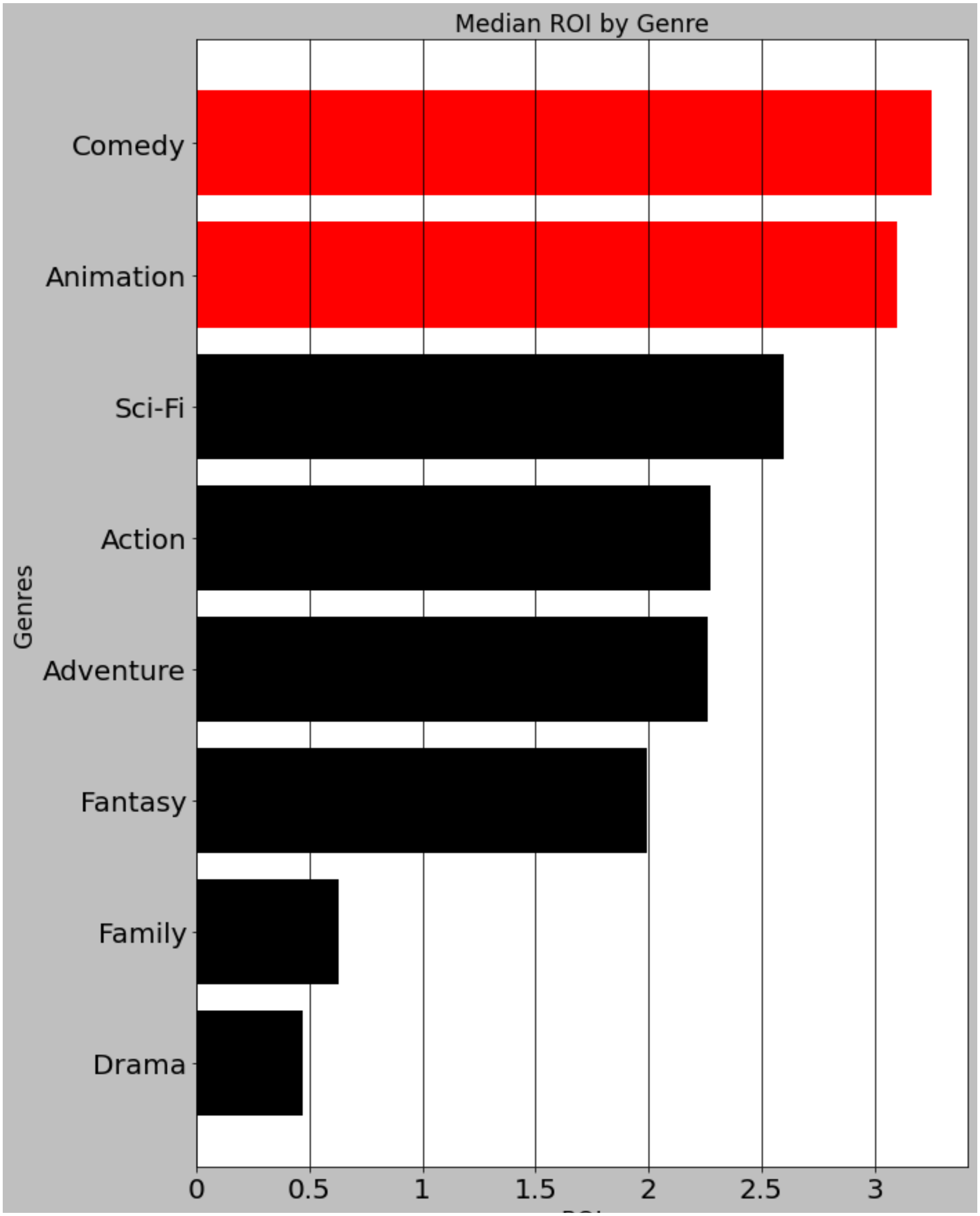
Out[431]:

```
genres
Drama        0.468726
Family       0.628989
Fantasy      1.995511
Adventure    2.261100
Action       2.276625
Sci-Fi       2.599421
Animation    3.101203
Comedy       3.250116
Name: roi, dtype: float64
```

In [432]:

```
#create a barplot of median roi per genre for high production cost
fig, ax = plt.subplots(figsize=(10,15))
y = roi_top_spend.index
width = roi_top_spend.sort_values()

ax.barh(y=y,width=width, color=['black','black','black','black','b
```

```
                                          'black','red','red'])
ax.set_title('Median ROI by Genre',fontsize='xx-large')
ax.set_xlabel('ROI', fontsize='xx-large')
ax.set_ylabel('Genres', fontsize='xx-large')
ax.tick_params(axis='both', which='major', labelsize=20)
ax.grid(axis='x')
ax.xaxis.set_major_formatter(tick.FuncFormatter(reformat_large_tic
```




**Analysis**

The chart shows that Comedy and Animation movies return the best ROI in movies which spend over $150M

In [433]:
```
#median profit of animated or comedy movies above $150M
data = top_spend.loc[top_spend['genres'].isin(['Comedy','Animatior
data.groupby('movie').median()['ww_profit'].median()
```

Out[433]: 501177456.0

The median profit for a comedy or animated movie is 501M dollars when spending over 150M dollars

In [434]:
```
#median production cost of animated or comedy movies above $150M
data.groupby('movie').median()['production_budget'].median()
```

Out[434]: 172500000.0

The median production cost for a comedy or animated movie is 172.5M dollars when spending over 150M dollars

In [435]:
```
#median profit of horror movies
data_2 = movies_df_genres.loc[movies_df_genres['genres'] =='Horror
data_2.groupby('movie').median()['ww_profit'].median()
```

Out[435]: 55989590.0

The median profit for a horror movie is 56M dollars

In [436]:
```
#median production cost of a horror movie
data_2.groupby('movie').median()['production_budget'].median()
```

Out[436]: 10000000.0

The median production cost for a horror movie is 10M dollars

In [437]:
```
movies_df
```

Out[437]:

| | tconst | start_year | runtime_minutes | genres | release_date | movie | proc |
|---|---|---|---|---|---|---|---|
| **0** | tt0359950 | 2013 | 114.0 | [Adventure, Comedy, Drama] | 2013-12-25 | The Secret Life of Walter Mitty | |
| **1** | tt0365907 | 2014 | 114.0 | [Action, Crime, Drama] | 2014-09-19 | A Walk Among the Tombstones | |
| **2** | tt0369610 | 2015 | 124.0 | [Action, Adventure, Sci-Fi] | 2015-06-12 | Jurassic World | |
| **3** | tt0376136 | 2011 | 119.0 | [Comedy, Drama] | 2011-10-28 | The Rum Diary | |
| **4** | tt0383010 | 2012 | 92.0 | [Comedy, Family] | 2012-04-13 | The Three Stooges | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **1016** | tt7334528 | 2018 | 103.0 | [Comedy, Sport] | 2018-06-29 | Uncle Drew | |
| **1017** | tt7349662 | 2018 | 135.0 | [Biography, Crime, Drama] | 2018-08-10 | BlacKkKlansman | |
| **1019** | tt7401588 | 2018 | 118.0 | [Comedy, Drama] | 2018-11-16 | Instant Family | |
| **1020** | tt7535780 | 2017 | 72.0 | [Documentary] | 2017-02-17 | The Great Wall | |
| **1022** | tt7959026 | 2018 | 116.0 | [Crime, Drama, Thriller] | 2018-12-14 | The Mule | |

945 rows × 16 columns

The median production budget for a comedy or animated movie is 172.5M dollars when spending over 150M dollars

## 5.3 Q3: Which genres have the highest chance of success?

I will define success of a movie where the roi is greater than or equal to 2 which is that the film has returned 2 times more profit than the initial cost.

In [438]:

```
#create a new column based off roi which measures success or failu
movies_df_5_3 = movies_df_genres.loc[:,['genres','roi']]
movies_df_5_3['Success'] = np.where(movies_df_5_3['roi'] >= 2, Tru
movies_df_5_3
```

Out[438]:

| | genres | roi | Success |
|---|---|---|---|
| **0** | Adventure | 1.064409 | False |
| **0** | Comedy | 1.064409 | False |
| **0** | Drama | 1.064409 | False |
| **1** | Action | 1.218164 | False |
| **1** | Crime | 1.218164 | False |
| **...** | ... | ... | ... |
| **1019** | Drama | 1.494504 | False |
| **1020** | Documentary | 1.229912 | False |
| **1022** | Crime | 2.417154 | True |
| **1022** | Drama | 2.417154 | True |
| **1022** | Thriller | 2.417154 | True |

2497 rows × 3 columns

In [439]:
```
#create success dataframe and calculating percentage success for
movies_df_5_3.loc[movies_df_5_2['Success'] == True]
```

Out[439]:

| | genres | roi | Success |
|---|---|---|---|
| **2** | Action | 6.669092 | True |
| **2** | Adventure | 6.669092 | True |
| **2** | Sci-Fi | 6.669092 | True |
| **9** | Horror | 4.923219 | True |
| **9** | Mystery | 4.923219 | True |
| **...** | ... | ... | ... |
| **1017** | Crime | 5.201156 | True |
| **1017** | Drama | 5.201156 | True |
| **1022** | Crime | 2.417154 | True |
| **1022** | Drama | 2.417154 | True |
| **1022** | Thriller | 2.417154 | True |

1088 rows × 3 columns

In [440]:
```
#calculate percentage successful for Horror, Thriller and Mystery
table_5_3 = pd.DataFrame(movies_df_5_3.groupby(by='genres').sum()[
table_5_3 = table_5_3.sort_values('Success').reset_index()
table_5_3.loc[table_5_3['genres'].isin(['Horror','Thriller','Myste
#used both 0 and 2 as roi thresholds in cell above to get numbers
```

Out[440]: 60.59648760005757

**Analysis**

Horror/Thriller/Mystery has a 61% chance of getting a ROI above 2

In [441]:
```
#calculating percentage successful for all other genres
table_5_3 = pd.DataFrame(movies_df_5_3.groupby(by='genres').sum()[
table_5_3 = table_5_3.sort_values('Success').reset_index()
table_5_3.loc[~table_5_3['genres'].isin(['Horror','Thriller','Myst
#used both 0 and 2 as roi thresholds in cell above to get numbers
```

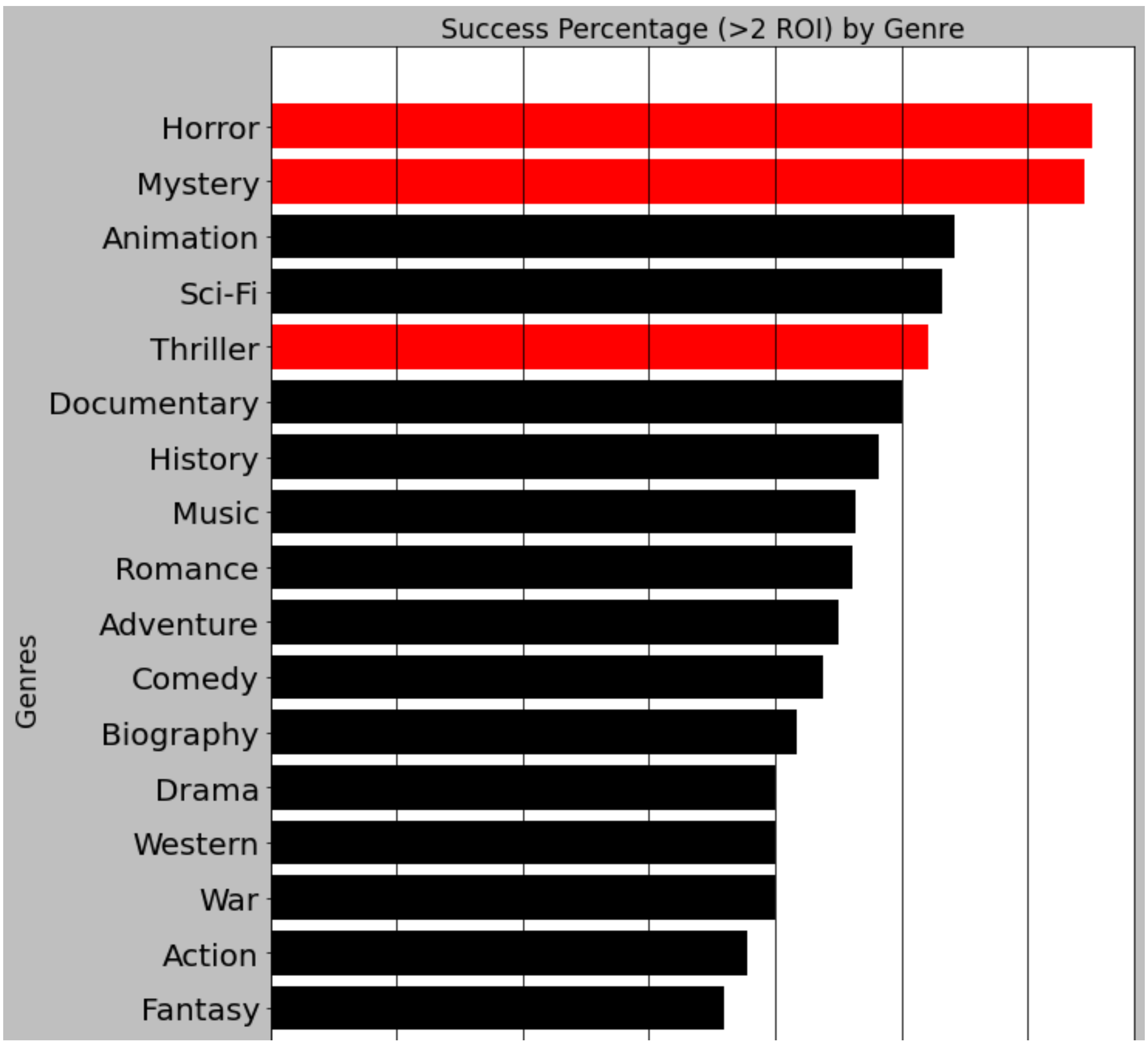
Out[441]: 40.74711937639691

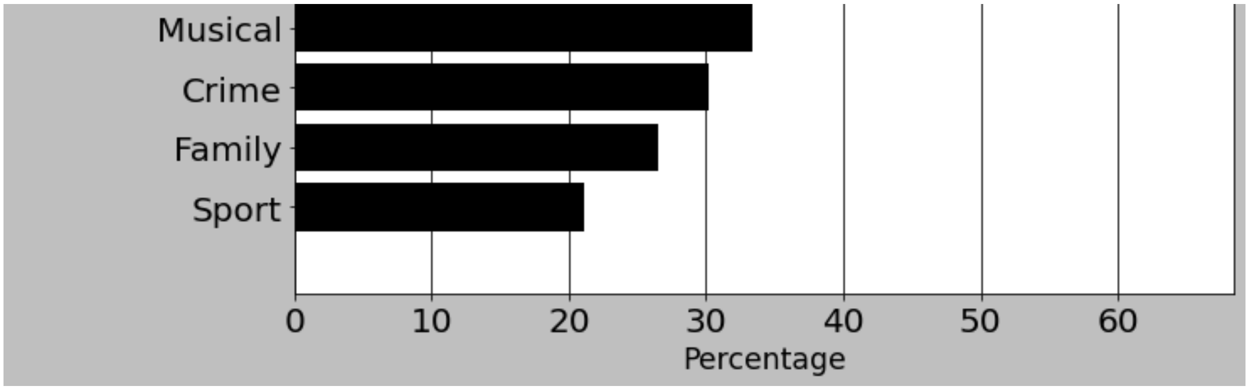
**Analysis**

All other genres have a 41% chance of getting a ROI above 2

In [442]:

```
#create a barplot of percentage success per genre
fig, ax = plt.subplots(figsize=(10,15))
y = table_5_3['genres']
width = table_5_3['Success']

ax.barh(y=y,width=width, color=['black','black','black','black','b
                                'black','black','black','black','b
                                'black','black','black','black','b
                                'black','black','red','red'])
ax.set_title('Success Percentage (>2 ROI) by Genre', fontsize='xx-
ax.set_xlabel('Percentage', fontsize='xx-large')
ax.set_ylabel('Genres', fontsize='xx-large')
ax.tick_params(axis='both', which='major', labelsize=20)
ax.grid(axis='x')
```






**Analysis**
Horror movies have a 61% chance of having a ROI of greater than or equal to 2. This is the highest chance of any genre with Mystery and Thriller being extremely high as well.

## 5.4 Q4: Which month should a Horror movie be released?

I want to know what months are successful Horror movies released

In [443]:

```
#look at dataframe
movies_df_genres
```

Out[443]:

| | tconst | start_year | runtime_minutes | genres | release_date | movie | productio |
|---|---|---|---|---|---|---|---|
| **0** | tt0359950 | 2013 | 114.0 | Adventure | 2013-12-25 | The Secret Life of Walter Mitty | |
| **0** | tt0359950 | 2013 | 114.0 | Comedy | 2013-12-25 | The Secret Life of Walter Mitty | |
| **0** | tt0359950 | 2013 | 114.0 | Drama | 2013-12-25 | The Secret Life of Walter Mitty | |
| **1** | tt0365907 | 2014 | 114.0 | Action | 2014-09-19 | A Walk Among the Tombstones | |
| **1** | tt0365907 | 2014 | 114.0 | Crime | 2014-09-19 | A Walk Among the Tombstones | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **1019** | tt7401588 | 2018 | 118.0 | Drama | 2018-11-16 | Instant Family | |
| **1020** | tt7535780 | 2017 | 72.0 | Documentary | 2017-02-17 | The Great Wall | 1 |
| **1022** | tt7959026 | 2018 | 116.0 | Crime | 2018-12-14 | The Mule | |
| **1022** | tt7959026 | 2018 | 116.0 | Drama | 2018-12-14 | The Mule | |
| **1022** | tt7959026 | 2018 | 116.0 | Thriller | 2018-12-14 | The Mule | |

2497 rows × 16 columns

In [444]:

```
#add success column
movies_df_genres['Success'] = np.where(movies_df_5_3['roi'] >= 2,
movies_df_genres
```

Out[444]:

| | tconst | start_year | runtime_minutes | genres | release_date | movie | productio |
|---|---|---|---|---|---|---|---|
| **0** | tt0359950 | 2013 | 114.0 | Adventure | 2013-12-25 | The Secret Life of Walter Mitty | |
| **0** | tt0359950 | 2013 | 114.0 | Comedy | 2013-12-25 | The Secret Life of Walter Mitty | |
| **0** | tt0359950 | 2013 | 114.0 | Drama | 2013-12-25 | The Secret Life of Walter Mitty | |
| **1** | tt0365907 | 2014 | 114.0 | Action | 2014-09-19 | A Walk Among the Tombstones | |
| **1** | tt0365907 | 2014 | 114.0 | Crime | 2014-09-19 | A Walk Among the Tombstones | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **1019** | tt7401588 | 2018 | 118.0 | Drama | 2018-11-16 | Instant Family | |
| **1020** | tt7535780 | 2017 | 72.0 | Documentary | 2017-02-17 | The Great Wall | 1 |
| **1022** | tt7959026 | 2018 | 116.0 | Crime | 2018-12-14 | The Mule | |
| **1022** | tt7959026 | 2018 | 116.0 | Drama | 2018-12-14 | The Mule | |
| **1022** | tt7959026 | 2018 | 116.0 | Thriller | 2018-12-14 | The Mule | |

2497 rows × 17 columns

In [445]:

```
#create a new dataframe with only horror movies which are successf
movies_df_horror = movies_df_genres.loc[(movies_df_genres['genres'
                                         'Horror') & (movies_df_ge
                                                     True)]

movies_df_horror
```

Out[445]:

| | tconst | start_year | runtime_minutes | genres | release_date | movie | production_b |
|---|---|---|---|---|---|---|---|
| **9** | tt0431021 | 2012 | 92.0 | Horror | 2012-08-31 | The Possession | 140 |
| **20** | tt0464154 | 2010 | 88.0 | Horror | 2010-08-20 | Piranha 3D | 240 |
| **38** | tt0498381 | 2017 | 102.0 | Horror | 2017-02-03 | Rings | 250 |
| **154** | tt1179933 | 2016 | 103.0 | Horror | 2016-03-11 | 10 Cloverfield Lane | 50 |
| **166** | tt1204977 | 2014 | 89.0 | Horror | 2014-10-24 | Ouija | 50 |
| **183** | tt1220634 | 2010 | 96.0 | Horror | 2010-09-10 | Resident Evil: Afterlife | 575 |
| **235** | tt1314655 | 2010 | 80.0 | Horror | 2010-09-17 | Devil | 100 |
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | The Last | [illegible] |

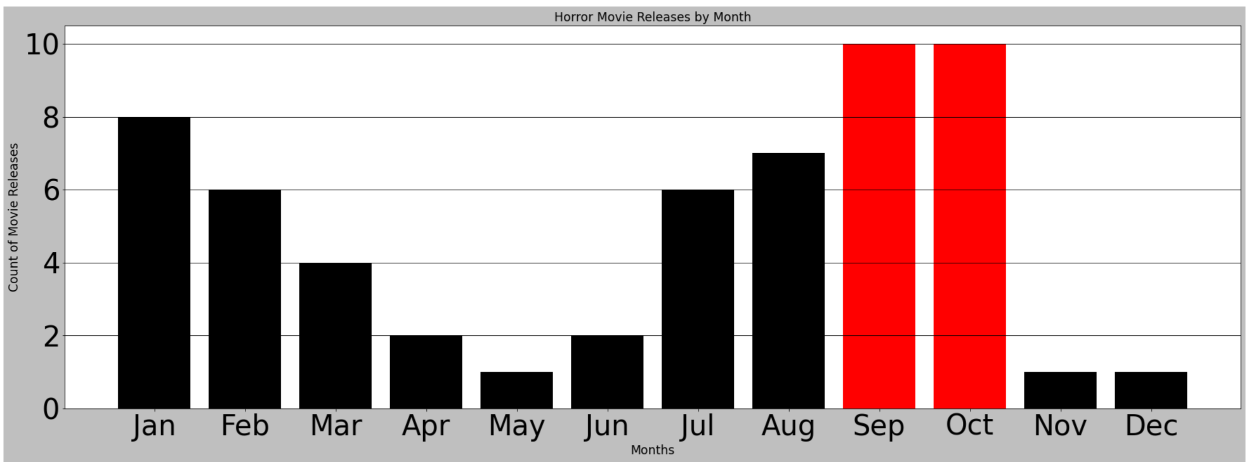
In [446]:
```
#create a dataframe for graphing
graph_release_month = movies_df_horror.groupby('release_month')['r


graph_release_month.reset_index(inplace=True)
month_dict = {1:'Jan',2:'Feb',3:'Mar',4:'Apr',5:'May',6:'Jun',7:'J
              9:'Sep',10:'Oct',11:'Nov',12:'Dec'}
graph_release_month.replace({'release_month':month_dict}, inplace=
graph_release_month
```

Out[446]:

| | release_month | movie_count | roi_median |
|---|---|---|---|
| **0** | Jan | 8 | 10.611761 |
| **1** | Feb | 6 | 6.793661 |
| **2** | Mar | 4 | 6.929399 |
| **3** | Apr | 2 | 22.428512 |
| **4** | May | 1 | 41.411721 |
| **5** | Jun | 2 | 18.778922 |
| **6** | Jul | 6 | 13.146386 |
| **7** | Aug | 7 | 4.410423 |
| **8** | Sep | 10 | 3.522098 |
| **9** | Oct | 10 | 25.782825 |
| **10** | Nov | 1 | 6.130898 |
| **11** | Dec | 1 | 3.119226 |

In [447]:

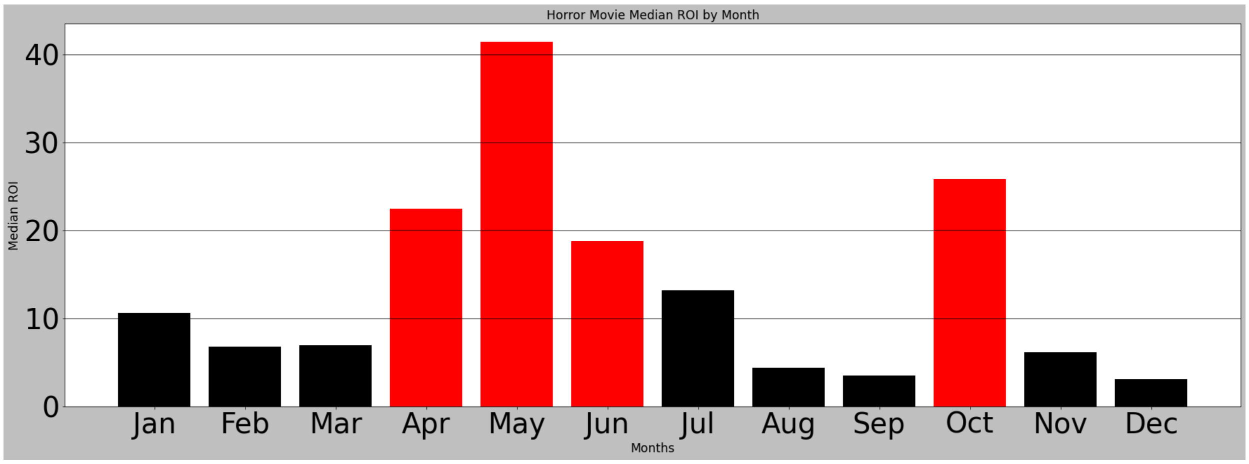
```
#create a barplot showing count of movie releases by month for Hor
fig, ax = plt.subplots(figsize=(30,10))
x=graph_release_month['release_month']
y=graph_release_month['movie_count']

ax.bar(x=x, height=y, color=['black', 'black','black', 'black','bl
                              'black', 'black','red', 'red','black

ax.set_title('Horror Movie Releases by Month', fontsize='xx-large'
ax.set_xlabel('Months', fontsize='xx-large')
ax.set_ylabel('Count of Movie Releases', fontsize='xx-large')
ax.tick_params(axis='both', which='major', labelsize=40)
ax.grid(axis='y')
```




### Analysis

This graph shows that the most frequent month that a horror movie gets releases is in September and October.

In [448]:
```
#create a barplot showing median roi of movie releases by month fo
fig, ax = plt.subplots(figsize=(30,10))
x=graph_release_month['release_month']
y=graph_release_month['roi_median']

ax.bar(x=x, height=y, color=['black', 'black','black', 'red','red'
                              'black', 'black','black', 'red','bla

ax.set_title('Horror Movie Median ROI by Month', fontsize='xx-larg
ax.set_xlabel('Months', fontsize='xx-large')
ax.set_ylabel('Median ROI', fontsize='xx-large')
ax.tick_params(axis='both', which='major', labelsize=40)
ax.grid(axis='y')
```




**Analysis**
This graph shows that the highest median ROI for a horror movie is in October. I believe releasing a movie in October would be ideal.

### 5.4.1 What Does Release Count and ROI Look Like For All Genres?

I want to see what the distribution of movie release count and roi are for each release month for all movies

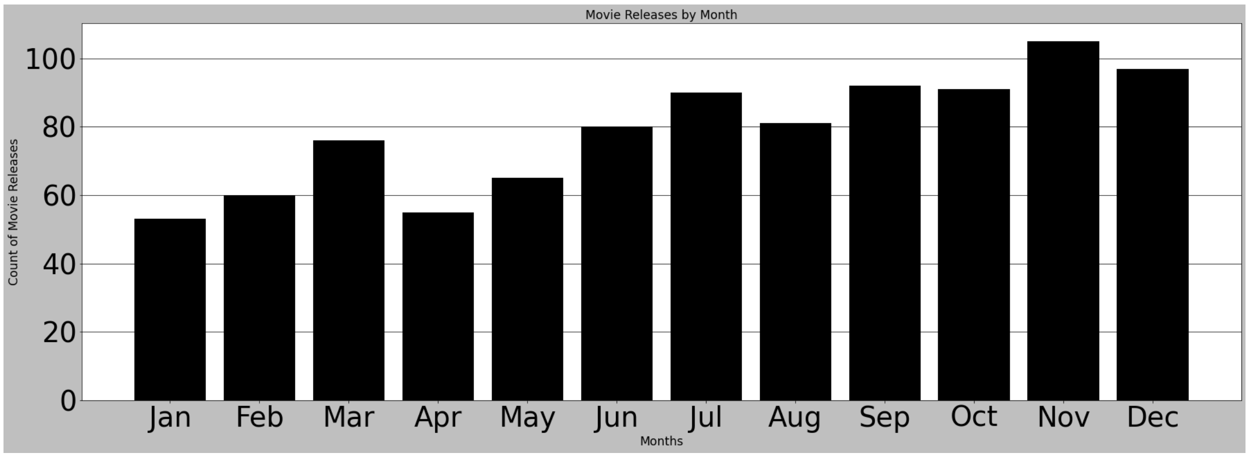
In [449]:
```
#create dataframe for graphs
graph_release_month_all = movies_df.groupby('release_month')['roi'


graph_release_month_all.reset_index(inplace=True)
month_dict = {1:'Jan',2:'Feb',3:'Mar',4:'Apr',5:'May',6:'Jun',7:'J
              9:'Sep',10:'Oct',11:'Nov',12:'Dec'}
graph_release_month_all.replace({'release_month':month_dict}, inpl
graph_release_month_all
```

Out[449]:

| | release_month | movie_count | roi_median |
|---|---|---|---|
| **0** | Jan | 53 | 1.383301 |
| **1** | Feb | 60 | 2.198686 |
| **2** | Mar | 76 | 1.335502 |
| **3** | Apr | 55 | 1.622187 |
| **4** | May | 65 | 1.771521 |
| **5** | Jun | 80 | 1.799111 |
| **6** | Jul | 90 | 2.232599 |
| **7** | Aug | 81 | 1.287002 |
| **8** | Sep | 92 | 1.482396 |
| **9** | Oct | 91 | 1.447092 |
| **10** | Nov | 105 | 1.932070 |
| **11** | Dec | 97 | 1.678453 |

In [450]:

```
#create a barplot showing count of movie releases by month for all
fig, ax = plt.subplots(figsize=(30,10))
x=graph_release_month_all['release_month']
y=graph_release_month_all['movie_count']

ax.bar(x=x, height=y, color=['black', 'black','black', 'black','bl
                             'black', 'black','black', 'black','b

ax.set_title('Movie Releases by Month', fontsize='xx-large')
ax.set_xlabel('Months', fontsize='xx-large')
ax.set_ylabel('Count of Movie Releases', fontsize='xx-large')
ax.tick_params(axis='both', which='major', labelsize=40)
ax.grid(axis='y')
```

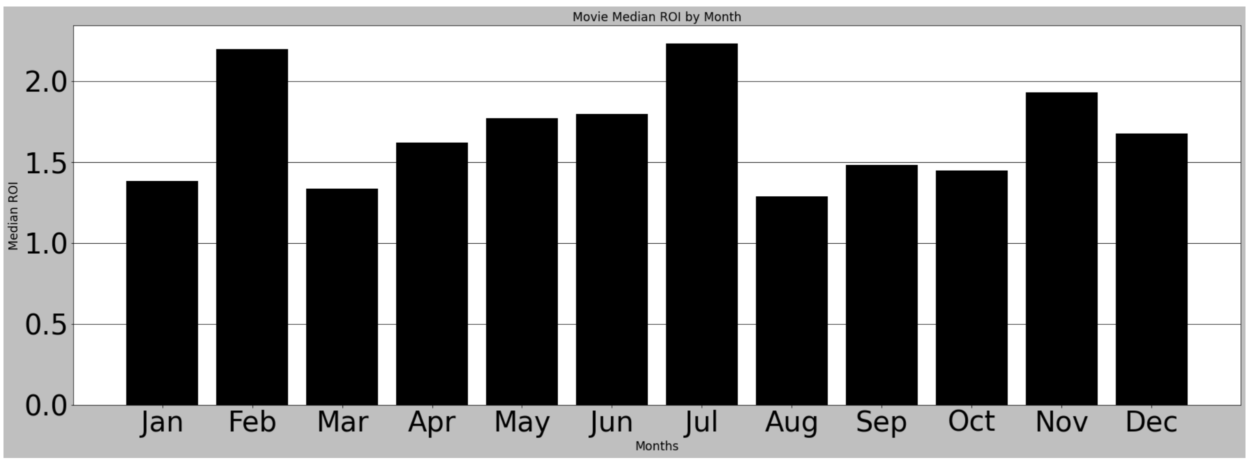



**Analysis**

The most common month for a movie release is in November

In [451]:

```
#create a barplot showing median roi of movie releases by month fo
fig, ax = plt.subplots(figsize=(30,10))
x=graph_release_month_all['release_month']
y=graph_release_month_all['roi_median']

ax.bar(x=x, height=y, color=['black', 'black','black', 'black','bl
                              'black', 'black','black', 'black','b

ax.set_title('Movie Median ROI by Month', fontsize='xx-large')
ax.set_xlabel('Months', fontsize='xx-large')
ax.set_ylabel('Median ROI', fontsize='xx-large')
ax.tick_params(axis='both', which='major', labelsize=40)
ax.grid(axis='y')
```




**Analysis**

The most successful month for ROI to release a movie is in July

## 5.5 Q5: How much should be spent on a Horror movie to get the best ROI?

I want to understand how production budget relates to roi for a horror film so that I can recommend a production budget range.

In [452]:

```
movies_df_horror
```

Out[452]:

| | tconst | start_year | runtime_minutes | genres | release_date | movie | production_b |
|---|---|---|---|---|---|---|---|
| 9 | tt0431021 | 2012 | 92.0 | Horror | 2012-08-31 | The Possession | 140 |
| 20 | tt0464154 | 2010 | 88.0 | Horror | 2010-08-20 | Piranha 3D | 240 |
| 38 | tt0498381 | 2017 | 102.0 | Horror | 2017-02-03 | Rings | 250 |
| 154 | tt1179933 | 2016 | 103.0 | Horror | 2016-03-11 | 10 Cloverfield Lane | 50 |
| 166 | tt1204977 | 2014 | 89.0 | Horror | 2014-10-24 | Ouija | 50 |
| 183 | tt1220634 | 2010 | 96.0 | Horror | 2010-09-10 | Resident Evil: Afterlife | 575 |
| 235 | tt1314655 | 2010 | 80.0 | Horror | 2010-09-17 | Devil | 100 |
| | | | | | | The Last | |

In [453]:

```
#create a dataframe for analysis
roi_vs_production_budget = movies_df_horror[['movie','production_b
                                                                  'roi','worldwide_gros
roi_vs_production_budget['ww_profit'] = roi_vs_production_budget['
roi_vs_production_budget
```

Out[453]:

| | movie | production_budget | roi | worldwide_gross | ww_profit |
|---|---|---|---|---|---|
| 9 | The Possession | 14000000 | 4.923219 | 82925064 | 68925064 |
| 20 | Piranha 3D | 24000000 | 2.485840 | 83660160 | 59660160 |
| 38 | Rings | 25000000 | 2.316691 | 82917283 | 57917283 |
| 154 | 10 Cloverfield Lane | 5000000 | 20.657284 | 108286422 | 103286422 |
| 166 | Ouija | 5000000 | 19.660126 | 103300632 | 98300632 |
| 183 | Resident Evil: Afterlife | 57500000 | 4.145638 | 295874190 | 238374190 |
| 235 | Devil | 10000000 | 5.335411 | 63354114 | 53354114 |
| 239 | The Last Exorcism | 1800000 | 37.981056 | 70165900 | 68365900 |
| 277 | When the Bough Breaks | 10000000 | 2.076845 | 30768449 | 20768449 |
| 283 | It | 35000000 | 18.927371 | 697457969 | 662457969 |
| 331 | The Conjuring | 20000000 | 14.900007 | 318000141 | 298000141 |

In [454]:

```
#create a list of conditions to bucket production_budget
conditions = [(roi_vs_production_budget['production_budget'] >0)
               & (roi_vs_production_budget['production_budget'] <=
              (roi_vs_production_budget['production_budget'] > 500
               & (roi_vs_production_budget['production_budget'] <=
              (roi_vs_production_budget['production_budget'] > 100
               & (roi_vs_production_budget['production_budget'] <=
              (roi_vs_production_budget['production_budget'] > 250

values = ['0-5M','5-10M','10-25M','25M+']

roi_vs_production_budget['cat'] = np.select(conditions,values)
roi_vs_production_budget
```

Out[454]:

| | movie | production_budget | roi | worldwide_gross | ww_profit | cat |
|---|---|---|---|---|---|---|
| **9** | The Possession | 14000000 | 4.923219 | 82925064 | 68925064 | 10-25M |
| **20** | Piranha 3D | 24000000 | 2.485840 | 83660160 | 59660160 | 10-25M |
| **38** | Rings | 25000000 | 2.316691 | 82917283 | 57917283 | 10-25M |
| **154** | 10 Cloverfield Lane | 5000000 | 20.657284 | 108286422 | 103286422 | 0-5M |
| **166** | Ouija | 5000000 | 19.660126 | 103300632 | 98300632 | 0-5M |
| **183** | Resident Evil: Afterlife | 57500000 | 4.145638 | 295874190 | 238374190 | 25M+ |
| **235** | Devil | 10000000 | 5.335411 | 63354114 | 53354114 | 5-10M |
| **239** | The Last | 1800000 | 37.981056 | 70165900 | 68365900 | 0-5M |

In [455]:

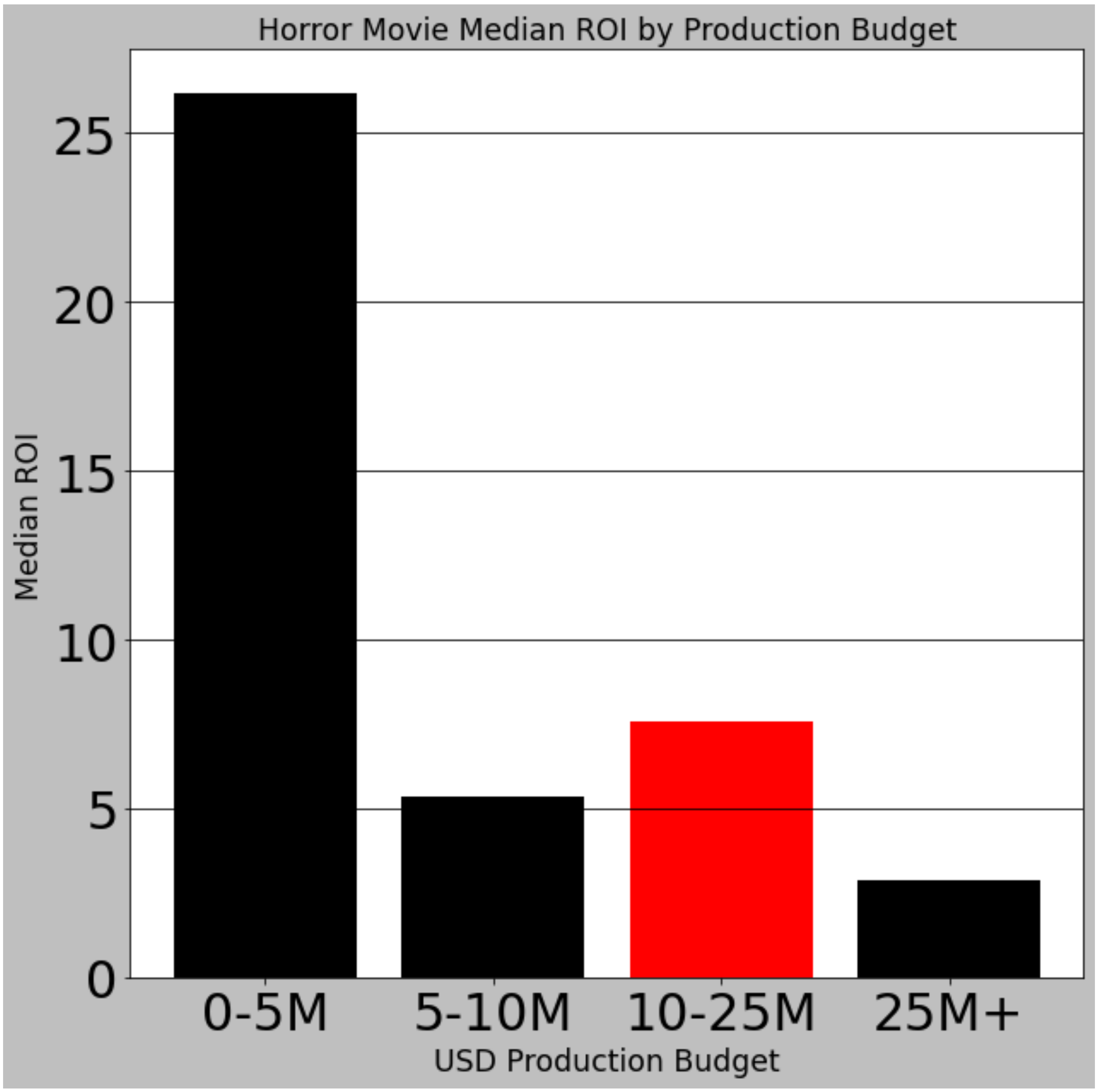
```
#creating dataframe to graph roi by quartile production budget
graph_roi_vs_prod_budget = roi_vs_production_budget.groupby('cat')
graph_roi_vs_prod_budget = graph_roi_vs_prod_budget.reindex(['0-5M
graph_roi_vs_prod_budget
```

Out[455]:

| cat | production_budget | roi | worldwide_gross | ww_profit |
|---|---|---|---|---|
| **0-5M** | 4000000 | 26.179241 | 91266581 | 88266581 |
| **5-10M** | 10000000 | 5.335411 | 54104225 | 44104225 |
| **10-25M** | 15000000 | 7.597060 | 118763442 | 105763442 |
| **25M+** | 40000000 | 2.875279 | 240647629 | 175647629 |

In [456]:

```
#create a barplot showing the median roi by production_budget quar
fig, ax = plt.subplots(figsize=(10,10))
x=graph_roi_vs_prod_budget.index
y=graph_roi_vs_prod_budget['roi']

ax.bar(x=x, height=y, color=['black','black','red','black'])

ax.set_title('Horror Movie Median ROI by Production Budget', fonts
ax.set_xlabel('USD Production Budget',fontsize='xx-large')
ax.set_ylabel('Median ROI',fontsize='xx-large')
ax.tick_params(axis='both', which='major', labelsize=30)
ax.grid(axis='y')
```



**Analysis**
This graph shows that ROI drops drastically the more you spend. Spending less than 5 million dollars will ensure a very high ROI. However, spending more will get you a still outstanding ROI compared to other genres.

### 5.5.1 Getting movie names and studios for examples of success.

In [457]:

```
#sort horror dataframe to get examples
movies_df_horror.sort_values('roi', ascending=False)
```

Out[457]:

| | tconst | start_year | runtime_minutes | genres | release_date | movie | production_b |
|---|---|---|---|---|---|---|---|
| **694** | tt2309260 | 2015 | 81.0 | Horror | 2015-07-10 | The Gallows | 1 |
| **384** | tt1560985 | 2012 | 83.0 | Horror | 2012-01-06 | The Devil Inside | 10 |
| **372** | tt1536044 | 2010 | 91.0 | Horror | 2010-10-20 | Paranormal Activity 2 | 30 |
| **960** | tt5052448 | 2017 | 104.0 | Horror | 2017-02-24 | Get Out | 50 |
| **605** | tt1991245 | 2012 | 86.0 | Horror | 2012-05-25 | Chernobyl Diaries | 10 |
| **524** | tt1778304 | 2011 | 83.0 | Horror | 2011-10-21 | Paranormal Activity 3 | 50 |
| **823** | tt3322940 | 2014 | 99.0 | Horror | 2014-10-03 | Annabelle | 65 |

## 5.6 Q6: Expected profit from a successful Horror film

I want to give Microsoft an understanding of how much money they can expect to make after investing around 5 million dollars on a Horror film.

In [458]:
```
#create profit column
movies_df_horror['ww_profit'] = movies_df_horror['worldwide_gross'
movies_df_horror
```

Out[458]:

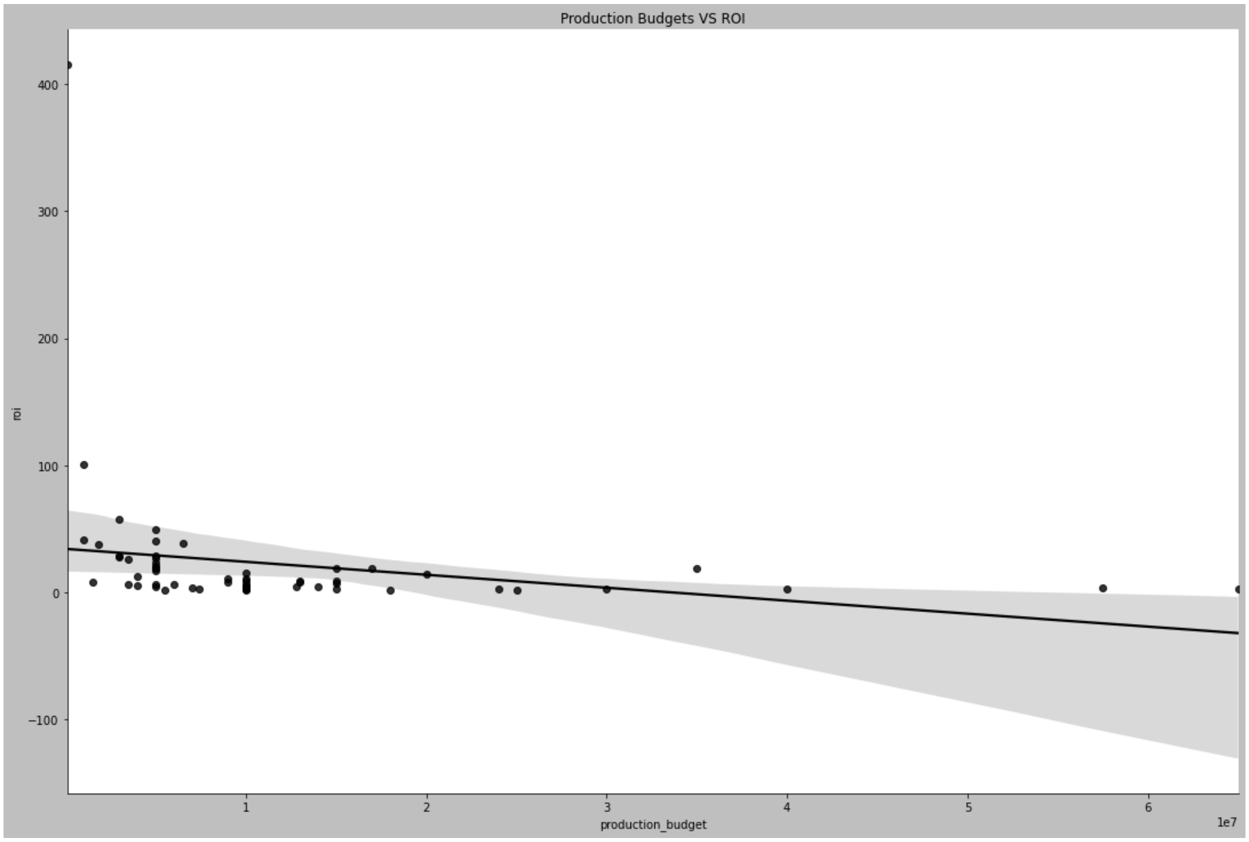
| | tconst | start_year | runtime_minutes | genres | release_date | movie | production_b |
|---|---|---|---|---|---|---|---|
| **9** | tt0431021 | 2012 | 92.0 | Horror | 2012-08-31 | The Possession | 140 |
| **20** | tt0464154 | 2010 | 88.0 | Horror | 2010-08-20 | Piranha 3D | 240 |
| **38** | tt0498381 | 2017 | 102.0 | Horror | 2017-02-03 | Rings | 250 |
| **154** | tt1179933 | 2016 | 103.0 | Horror | 2016-03-11 | 10 Cloverfield Lane | 50 |
| **166** | tt1204977 | 2014 | 89.0 | Horror | 2014-10-24 | Ouija | 50 |
| **183** | tt1220634 | 2010 | 96.0 | Horror | 2010-09-10 | Resident Evil: Afterlife | 575 |
| **235** | tt1314655 | 2010 | 80.0 | Horror | 2010-09-17 | Devil | 100 |
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | The Last | [illegible] |

In [459]:
```
#get the median ww profit for a successful horror movie
movies_df_horror['ww_profit'].median()
```

Out[459]: 85513231.0

In [460]:
```
#create a lmplot showing the correlation between production budget
sns.lmplot(x='production_budget', y='roi', data=roi_vs_production_

```

Out[460]: <seaborn.axisgrid.FacetGrid at 0x7fbe3a5ba790>




## 5.7 Q8: How does horror ratings relate to revenue?

I want to understand how spending relates to the popularity of a horror film

In [461]:
```
#grab starting dataframe
movies_df_genres
```

Out[461]:

| | tconst | start_year | runtime_minutes | genres | release_date | movie | productio |
|---|---|---|---|---|---|---|---|
| **0** | tt0359950 | 2013 | 114.0 | Adventure | 2013-12-25 | The Secret Life of Walter Mitty | |
| **0** | tt0359950 | 2013 | 114.0 | Comedy | 2013-12-25 | The Secret Life of Walter Mitty | |
| **0** | tt0359950 | 2013 | 114.0 | Drama | 2013-12-25 | The Secret Life of Walter Mitty | |
| **1** | tt0365907 | 2014 | 114.0 | Action | 2014-09-19 | A Walk Among the Tombstones | |
| **1** | tt0365907 | 2014 | 114.0 | Crime | 2014-09-19 | A Walk Among the Tombstones | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **1019** | tt7401588 | 2018 | 118.0 | Drama | 2018-11-16 | Instant Family | |
| **1020** | tt7535780 | 2017 | 72.0 | Documentary | 2017-02-17 | The Great Wall | 1 |
| **1022** | tt7959026 | 2018 | 116.0 | Crime | 2018-12-14 | The Mule | |
| **1022** | tt7959026 | 2018 | 116.0 | Drama | 2018-12-14 | The Mule | |
| **1022** | tt7959026 | 2018 | 116.0 | Thriller | 2018-12-14 | The Mule | |

2497 rows × 17 columns

In [462]:
```
#create new dataframe
horror_rtgs_trends = movies_df_genres.loc[movies_df_genres['genres
horror_rtgs_trends
```

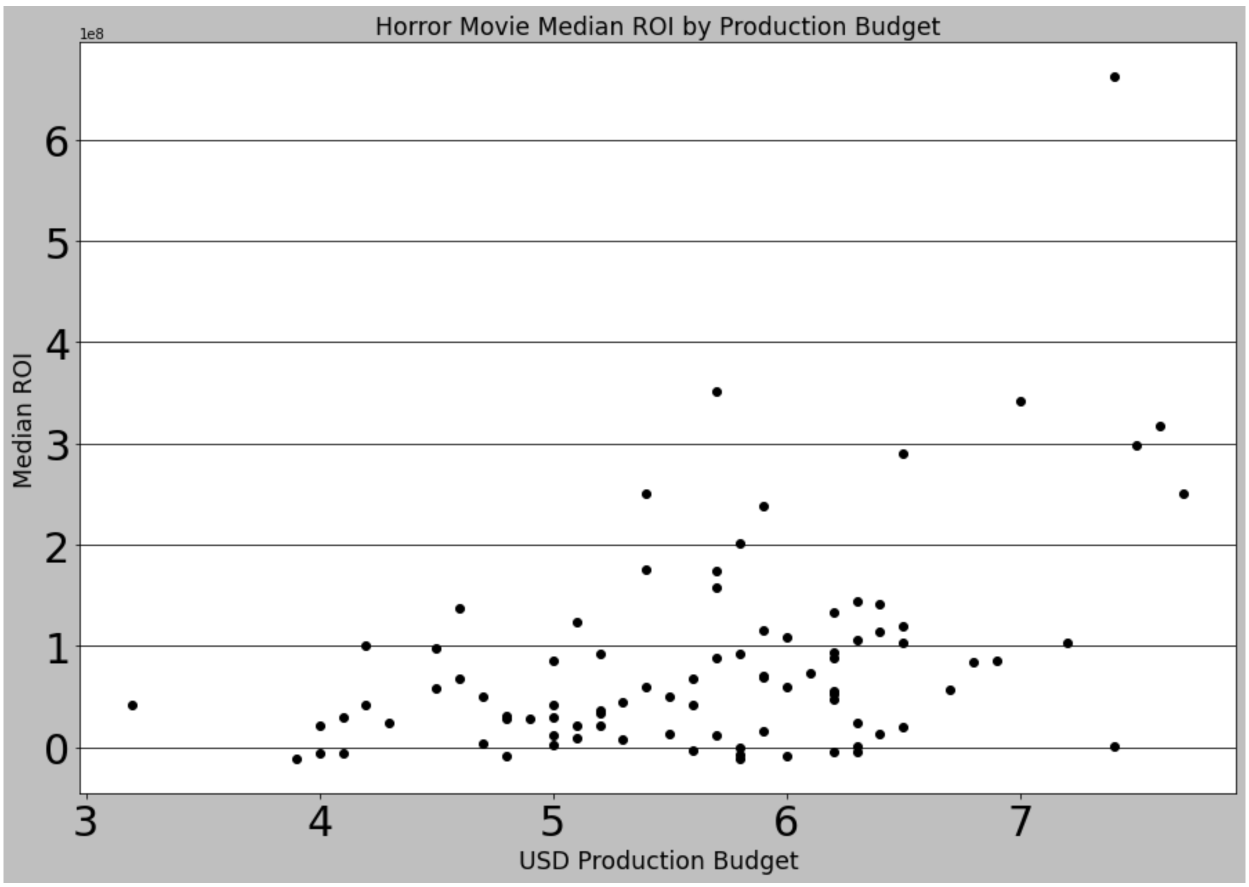
Exorcism

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| rror | 2010-06-18 | Cyrus | 7000000 | 7468936 | 10062896 | 4.7 | 94 |
| rror | 2016-02-05 | Pride and Prejudice and Zombies | 28000000 | 10907291 | 16638300 | 5.8 | 4618 |
| rror | 2016-09-09 | When the Bough Breaks | 10000000 | 29747603 | 30768449 | 5.1 | 472 |
| rror | 2017-09-08 | It | 35000000 | 327481748 | 697457969 | 7.4 | 35912 |
| rror | 2014-01-24 | I, Frankenstein | 65000000 | 19075290 | 74575290 | 5.1 | 7491 |
| rror | 2012-08-24 | The Apparition | 17000000 | 4936819 | 10637281 | 4.1 | 1811 |
| rror | 2013-07-19 | The Conjuring | 20000000 | 137400141 | 318000141 | 7.5 | 39723 |
| rror | 2011-03-11 | Red Riding Hood | 42000000 | 37662162 | 91678442 | 5.5 | 10236 |

The Vatican

In [463]:

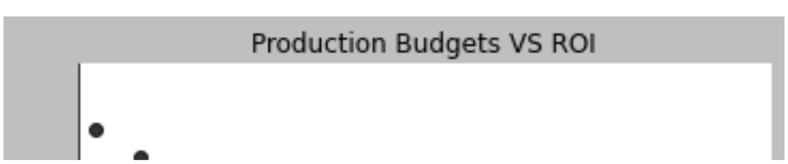
```
#create scatter plot of average rating, ww profit
fig, ax = plt.subplots(figsize=(15,10))
x=horror_rtgs_trends['averagerating']
y=horror_rtgs_trends['ww_profit']

ax.scatter(x=x, y=y)

ax.set_title('Horror Movie Median ROI by Production Budget', fonts
ax.set_xlabel('USD Production Budget',fontsize='xx-large')
ax.set_ylabel('Median ROI',fontsize='xx-large')
ax.tick_params(axis='both', which='major', labelsize=30)
ax.grid(axis='y')
```

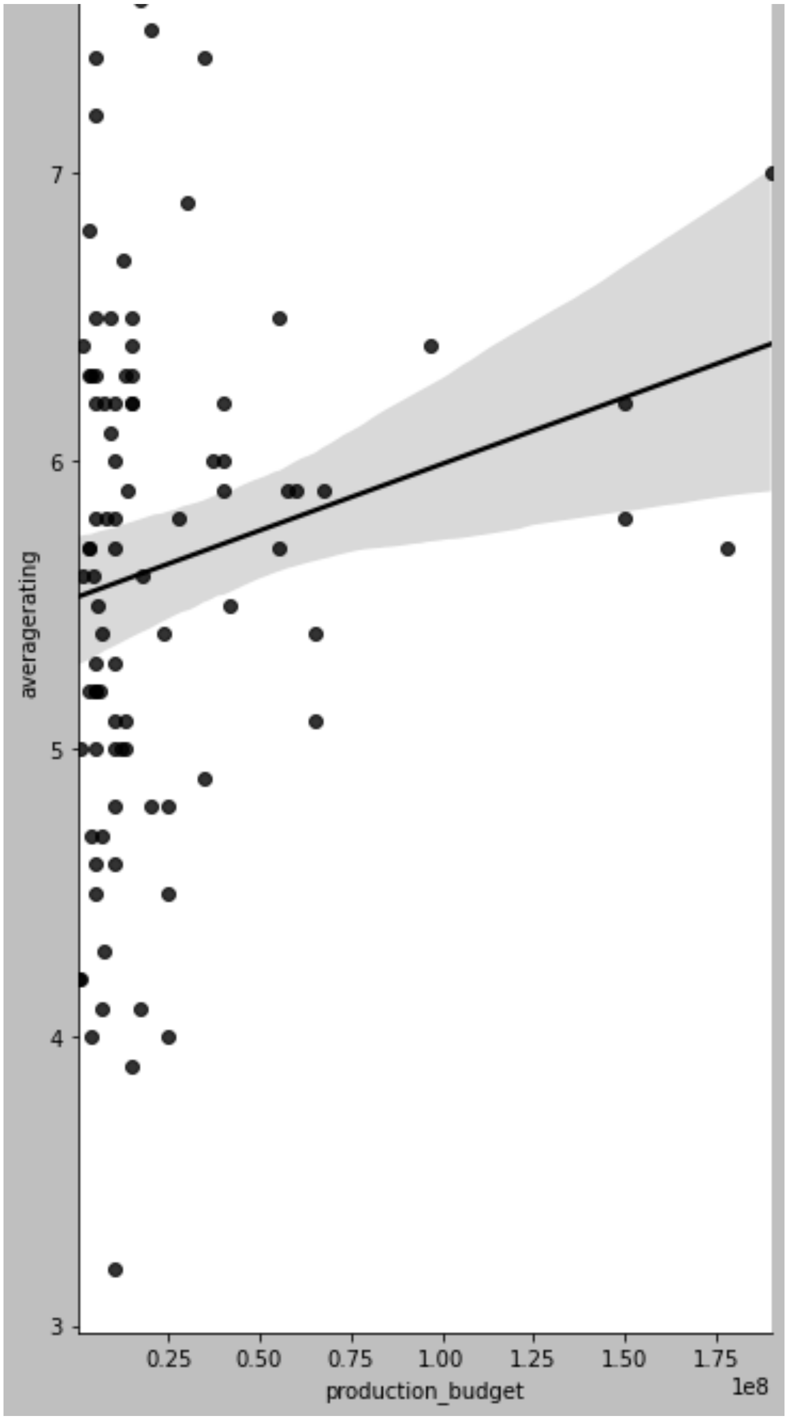



In [464]:

```
#create a lmplot showing the correlation between ratings and ww pr
sns.lmplot(x='production_budget', y='averagerating', data=horror_r
```

Out[464]: <seaborn.axisgrid.FacetGrid at 0x7fbe91b64820>






**Analysis**
The scatter plot shows that there is a positive correlation between spending more on a horror film and the average rating. I will recommend to spend a bit more on the movie.

# 6 Appendix

Things I explored but aren't part of recommendations

## 6.1 Q7:What director should be considered?

I want to see if there has been a director who stands out as producing the best film as it pertains to experience, average movie rating and ROI.

I will be joining these tables on:

- movies_df: [tconst]
- imdb_title_principals: [tconst]

In [465]:
```
#join dataframe 1
movies_df_directors = movies_df_genres.merge(imdb_title_principals
movies_df_directors
```

Out[465]:

| | tconst | start_year | runtime_minutes | genres | release_date | movie | production_budg |
|---|---|---|---|---|---|---|---|
| **0** | tt0359950 | 2013 | 114.0 | Adventure | 2013-12-25 | The Secret Life of Walter Mitty | 910000 |
| **1** | tt0359950 | 2013 | 114.0 | Adventure | 2013-12-25 | The Secret Life of Walter Mitty | 910000 |
| **2** | tt0359950 | 2013 | 114.0 | Adventure | 2013-12-25 | The Secret Life of Walter Mitty | 910000 |
| **3** | tt0359950 | 2013 | 114.0 | Adventure | 2013-12-25 | The Secret Life of Walter Mitty | 910000 |
| **4** | tt0359950 | 2013 | 114.0 | Adventure | 2013-12-25 | The Secret Life of Walter Mitty | 910000 |
| **...** | ... | ... | ... | ... | ... | ... | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **24932** | tt7959026 | 2018 | 116.0 | Thriller | 2018-12-14 | The Mule | 500000 |
| **24933** | tt7959026 | 2018 | 116.0 | Thriller | 2018-12-14 | The Mule | 500000 |
| **24934** | tt7959026 | 2018 | 116.0 | Thriller | 2018-12-14 | The Mule | 500000 |
| **24935** | tt7959026 | 2018 | 116.0 | Thriller | 2018-12-14 | The Mule | 500000 |
| **24936** | tt7959026 | 2018 | 116.0 | Thriller | 2018-12-14 | The Mule | 500000 |

24937 rows × 22 columns

I will be joining these tables on:

- movies_df: [nconst]
- imdb_name_basics: [nconst]

In [466]:

```
#join dataframe 2
movies_df_directors = movies_df_directors.merge(imdb_name_basics,
movies_df_directors
```

Out[466]:

| | tconst | start_year | runtime_minutes | genres | release_date | movie | production_bud |
|---|---|---|---|---|---|---|---|
| **0** | tt0359950 | 2013 | 114.0 | Adventure | 2013-12-25 | The Secret Life of Walter Mitty | 91000 |
| **1** | tt0359950 | 2013 | 114.0 | Comedy | 2013-12-25 | The Secret Life of Walter Mitty | 91000 |
| **2** | tt0359950 | 2013 | 114.0 | Drama | 2013-12-25 | The Secret Life of Walter Mitty | 91000 |
| **3** | tt1430626 | 2012 | 88.0 | Adventure | 2012-04-27 | The Pirates! Band | 55000 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | of Misfits | |
| **4** | tt1430626 | 2012 | 88.0 | Animation | 2012-04-27 | The Pirates! Band of Misfits | 55000 |
| **...** | ... | ... | ... | ... | ... | ... | |
| **24932** | tt7959026 | 2018 | 116.0 | Drama | 2018-12-14 | The Mule | 50000 |
| **24933** | tt7959026 | 2018 | 116.0 | Thriller | 2018-12-14 | The Mule | 50000 |
| **24934** | tt7959026 | 2018 | 116.0 | Crime | 2018-12-14 | The Mule | 50000 |
| **24935** | tt7959026 | 2018 | 116.0 | Drama | 2018-12-14 | The Mule | 50000 |
| **24936** | tt7959026 | 2018 | 116.0 | Thriller | 2018-12-14 | The Mule | 50000 |

24937 rows × 27 columns

In [467]:
```
#filter horror movies
movies_df_directors = movies_df_directors.loc[movies_df_directors[
movies_df_directors['category'].value_counts()
```

Out[467]:
```
actor                  200
producer               196
writer                 164
actress                156
director                93
composer                40
cinematographer         25
editor                  10
production_designer      5
Name: category, dtype: int64
```

In [468]:

```
#filter directors
movies_df_directors = movies_df_directors.loc[movies_df_directors[
movies_df_directors
```

Out[468]:

| | tconst | start_year | runtime_minutes | genres | release_date | movie | produc |
|---|---|---|---|---|---|---|---|
| **763** | tt0431021 | 2012 | 92.0 | Horror | 2012-08-31 | The Possession | |
| **778** | tt1204977 | 2014 | 89.0 | Horror | 2014-10-24 | Ouija | |
| **1539** | tt0780653 | 2010 | 103.0 | Horror | 2010-02-12 | The Wolfman | |
| **1638** | tt0464154 | 2010 | 88.0 | Horror | 2010-08-20 | Piranha 3D | |
| **2607** | tt5690360 | 2018 | 93.0 | Horror | 2018-08-10 | Slender Man | |
| **3138** | tt0498381 | 2017 | 102.0 | Horror | 2017-02-03 | Rings | |
| **4494** | tt0816711 | 2013 | 116.0 | Horror | 2013-06-21 | World War Z | |
| **4659** | tt2387433 | 2013 | 97.0 | Horror | 2013-02-22 | Dark Skies | |
| **5108** | tt0872230 | 2010 | 107.0 | Horror | 2010-10-08 | My Soul to Take | |
| **5111** | tt1262416 | 2011 | 111.0 | Horror | 2011-04-15 | Scream 4 | |

In [469]:

```
#looking at different directors
movies_df_directors.loc[movies_df_directors['primary_name'] == 'Ch
```

Out[469]:

| | tconst | start_year | runtime_minutes | genres | release_date | movie | production_bu |
|---|---|---|---|---|---|---|---|
| **16183** | tt1727776 | 2015 | 93.0 | Horror | 2015-10-30 | Scouts Guide to the Zombie Apocalypse | 1500 |
| **16190** | tt2473682 | 2014 | 84.0 | Horror | 2014-01-03 | Paranormal Activity: The Marked Ones | 500 |
| **16193** | tt5308322 | 2017 | 96.0 | Horror | 2017-10-13 | Happy Death Day | 500 |

3 rows × 27 columns

In [470]:
```
#determine count of movies, median roi and averagerating by direct
movie_count = pd.DataFrame(movies_df_directors.groupby('primary_na
movie_count['roi'] = movies_df_directors.groupby('primary_name').m
movie_count['averagerating'] = movies_df_directors.groupby('primar
movie_count.sort_values('averagerating',ascending=False)
```

Out[470]:

| primary_name | movie | roi | averagerating |
|---|---|---|---|
| **Jordan Peele** | 1 | 50.073590 | 7.70 |
| **James Wan** | 1 | 14.900007 | 7.50 |
| **Jeff Nichols** | 1 | 0.046740 | 7.40 |
| **Dan Trachtenberg** | 1 | 20.657284 | 7.20 |
| **Marc Forster** | 1 | 1.797446 | 7.00 |
| **Jonathan Levine** | 1 | 2.837387 | 6.90 |
| **Andy Muschietti** | 2 | 13.900204 | 6.80 |
| **Scott Derrickson** | 1 | 28.242602 | 6.80 |
| **Brad Anderson** | 1 | 4.454803 | 6.70 |
| **Guillermo del Toro** | 1 | 0.363034 | 6.50 |

**Analysis**
It is hard to determine a specific director to target since most directors have only directed a single movie. I will not recommend a specific director.

## 6.2 Percentage of Movies that do not make their money back

INTERESTING FACT: 36% of movies do not make any profit when using all data in tn_movie_budgets and 28% when using the tn_movie_budgets when joined with other tables. This is 15% when look at only the top 40 studios.