Phase 1 Project: Microsoft Enters the Movie Industry

Data Import

In [540]:

```
# Import standard packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import Image

%matplotlib inline
```

In [541]:

```
#rotten tomatoes databases
rt_reviews = pd.read_csv('data/zippedData/rotten_tomatoes_critic_reviews.csv.gz')
rt_movies = pd.read_csv('data/zippedData/rotten_tomatoes_movies.csv.gz')

#imdb databases
imdb_ratings = pd.read_csv('data/zippedData/imdb.title.ratings.csv.gz')
imdb_basics = pd.read_csv('data/zippedData/imdb.title.basics.csv.gz')
imdb_name = pd.read_csv('data/zippedData/imdb.name.basics.csv.gz')
akas = pd.read_csv('data/zippedData/imdb.title.akas.csv.gz')
crew = pd.read_csv('data/zippedData/imdb.title.crew.csv.gz')
principals = pd.read_csv('data/zippedData/imdb.title.principals.csv.gz')

# Box Office Mojo database
gross = pd.read_csv('data/zippedData/bom.movie_gross.csv.gz')

#the-numbers Database
budgets = pd.read_csv('data/zippedData/tn.movie_budgets.csv.gz')

#TheMovieDB.org
tmdb = pd.read_csv('data/zippedData/tmdb.movies.csv.gz')
```

In [542]:

```
def df_info(df):
    ''' Takes in a pandas dataframe (df) and returns len(df), df.isna()sum, df.shape, df.in
    print('#################################################################################
    name =[x for x in globals() if globals()[x] is df][0]
    print('{} Database'.format(name))
    print('\n')
    print('The length of this dataframe is {}'.format(len(df)))
    print('Database shape {}'.format(df.shape))
    print('\n')
    print(df.info())
    display(df.head())
    print('Number of Null values in {}'.format((name).capitalize()))
    display(df.isna().sum())
    print('The Number of duplicated rows in {} is {}'.format(name, df.duplicated().sum()))
    print('#################################################################################
```

Data Preview

It would now be useful to preview each dataset, this will determine which datasets I will proceed with and ones which I would recommend revisiting for further insights

In [543]:

```
df_info(imdb_basics)
```

```
################################################################################
########

imdb_basics Database


The length of this dataframe is 146144
Database shape (146144, 6)


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   tconst           146144 non-null  object
 1   primary_title    146144 non-null  object
 2   original_title   146123 non-null  object
 3   start_year       146144 non-null  int64
 4   runtime_minutes  114405 non-null  float64
 5   genres           140736 non-null  object
dtypes: float64(1), int64(1), object(4)
memory usage: 6.7+ MB
None
```

	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

```
Number of Null values in Imdb_basics

tconst                0
primary_title         0
original_title       21
start_year            0
runtime_minutes    31739
genres              5408
dtype: int64

The Number of duplicated rows in imdb_basics is 0
################################################################################
##########
```

This is a large dataset of over 146,000 movies, the "tconst" column will prove useful when merging and joining with other IMDB databases. This dataset will be used in further analysis.

Original title seems superfluous and has 21 null values out of 146144 entries. This column could be dropped and we will work with the primary_title column

runtime_minutes has a high % of null values - we may need to impute or remove these rows entirely, imputing in such a high number of rows could lead to erroneous conclusions being drawn from analysis

Cleaning recommendations

- Remove 'original_title' column
- Impute values into 'runtime_minutes' rows with null values or remove rows altogether
- Remove rows with null values in 'genres'

In [544]:

```
df_info(akas)
```

```
################################################################################
########

akas Database


The length of this dataframe is 331703
Database shape (331703, 8)


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 331703 entries, 0 to 331702
Data columns (total 8 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   title_id           331703 non-null  object 
 1   ordering           331703 non-null  int64  
 2   title              331703 non-null  object 
 3   region             278410 non-null  object 
 4   language           41715 non-null   object 
 5   types              168447 non-null  object 
 6   attributes         14925 non-null   object 
 7   is_original_title  331678 non-null  float64
dtypes: float64(1), int64(1), object(6)
memory usage: 20.2+ MB
None
```

	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

```
Number of Null values in Akas

title_id                 0
ordering                 0
title                    0
region               53293
language            289988
types               163256
attributes          316778
is_original_title       25
dtype: int64

The Number of duplicated rows in akas is 0
################################################################################
```

```
########
```

Messy dataset that doesn't appear to contain anything useful as first glance. The regions column could potentially be used to correlate 'number of regions' vs profit or return on investment but even the 'region' column contains a high number of null values. For now this dataset won't be used.

In [545]:

```
df_info(crew)
```

```
################################################################################
########

crew Database


The length of this dataframe is 146144
Database shape (146144, 3)


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 3 columns):
 #   Column     Non-Null Count   Dtype 
---  ------     --------------   ----- 
 0   tconst     146144 non-null  object
 1   directors  140417 non-null  object
 2   writers    110261 non-null  object
dtypes: object(3)
memory usage: 3.3+ MB
None
```

	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

```
Number of Null values in Crew

tconst           0
directors     5727
writers      35883
dtype: int64

The Number of duplicated rows in crew is 0
################################################################################
########
```

This dataframe is the same length as imdb_basics and also contains 'tconst'. The writers contains a high number of null values, I will not drop these rows because we will lose valuable information in the 'directors'

column. Instead I will keep writers and see how useful it proves to be later on i.e how many of the more successful movies have missing values in 'writers' column and can it be used to draw insights.

Cleaning Recommendations

- Merge with imdb_basics
- drop rows with null values in 'directors'

In [546]:

```
df_info(principals)
```

```
################################################################################
########

principals Database


The length of this dataframe is 1028186
Database shape (1028186, 6)


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1028186 entries, 0 to 1028185
Data columns (total 6 columns):
 #   Column      Non-Null Count    Dtype
---  ------      --------------    -----
 0   tconst      1028186 non-null  object
 1   ordering    1028186 non-null  int64
 2   nconst      1028186 non-null  object
 3   category    1028186 non-null  object
 4   job         177684 non-null   object
 5   characters  393360 non-null   object
dtypes: int64(1), object(5)
memory usage: 47.1+ MB
None
```

	tconst	ordering	nconst	category	job	characters
0	tt0111414	1	nm0246005	actor	NaN	["The Man"]
1	tt0111414	2	nm0398271	director	NaN	NaN
2	tt0111414	3	nm3739909	producer	producer	NaN
3	tt0323808	10	nm0059247	editor	NaN	NaN
4	tt0323808	1	nm3579312	actress	NaN	["Beth Boothby"]

```
Number of Null values in Principals

tconst             0
ordering           0
nconst             0
category           0
job           850502
characters    634826
dtype: int64

The Number of duplicated rows in principals is 0
################################################################################
########
```

Over 1million entries in this database, each movie ('tconst') has multiple entries for each individual 'nconst' which is the unique code given to individuals. We will need to see if there is a way to match up 'nconst' with a name - nconst alone isn't very useful. Job and characters columns are not required nor is ordering.

Cleaning recommendations

- Drop 'ordering' , 'job' and 'characters' columns

In [547]:

```
df_info(imdb_ratings)
```

```
################################################################################
########

imdb_ratings Database


The length of this dataframe is 73856
Database shape (73856, 3)


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73856 entries, 0 to 73855
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   tconst         73856 non-null  object
 1   averagerating  73856 non-null  float64
 2   numvotes       73856 non-null  int64
dtypes: float64(1), int64(1), object(1)
memory usage: 1.7+ MB
None
```

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

```
Number of Null values in Imdb_ratings

tconst           0
averagerating    0
numvotes         0
dtype: int64

The Number of duplicated rows in imdb_ratings is 0
################################################################################
########
```

This will be a useful dataset when it comes to looking for a correlation between 'averagerating' and profit or ROI. It is noted this dataframe is shorter than other IMDB datasets, therefore any analysis involving average rating will be on a lower number of movies.

Cleaning recommendations

- 'averagerating' will be influenced by 'numvotes' - insights drawn from 'averagerating' that have low number of votes may not be too valuable. It may be worth applying a filter (minimum number of votes) to this dataset.

In [548]:

```
df_info(imdb_name)
```

```
################################################################################
########

imdb_name Database


The length of this dataframe is 606648
Database shape (606648, 6)


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 606648 entries, 0 to 606647
Data columns (total 6 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   nconst              606648 non-null  object
 1   primary_name        606648 non-null  object
 2   birth_year          82736 non-null   float64
 3   death_year          6783 non-null    float64
 4   primary_profession  555308 non-null  object
 5   known_for_titles    576444 non-null  object
dtypes: float64(2), object(4)
memory usage: 27.8+ MB
None
```

	nconst	primary_name	birth_year	death_year	primary_profe
0	nm0061671	Mary Ellen Bauder	NaN	NaN	miscellaneous,production_manager,pro
1	nm0061865	Joseph Bauer	NaN	NaN	composer,music_department,sound_depar
2	nm0062070	Bruce Baum	NaN	NaN	miscellaneous,actor,
3	nm0062195	Axel Baumann	NaN	NaN	camera_department,cinematographer,art_depar
4	nm0062798	Pete Baxter	NaN	NaN	production_designer,art_department,set_dec

```
Number of Null values in Imdb_name

nconst                   0
primary_name             0
birth_year          523912
death_year          599865
primary_profession   51340
known_for_titles     30204
dtype: int64

The Number of duplicated rows in imdb_name is 0
################################################################################
########
```

This dataset will be useful for matching up 'nconst' with a name in the 'principals' dataframe.

Cleaning recommendations

- Drop 'birth_year' and 'death_year' columns
- 'primary_profession' appears to contain the same information as 'category' in 'principals' dataframe. This could be dropped.
- merge with principals dataframe

In [549]:

```
df_info(rt_movies)
```

```
################################################################################
########

rt_movies Database


The length of this dataframe is 17712
Database shape (17712, 22)


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17712 entries, 0 to 17711
Data columns (total 22 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   rotten_tomatoes_link            17712 non-null  object
 1   movie_title                     17712 non-null  object
 2   movie_info                      17391 non-null  object
 3   critics_consensus               9134 non-null   object
 4   content_rating                  17712 non-null  object
 5   genres                          17693 non-null  object
 6   directors                       17518 non-null  object
 7   authors                         16170 non-null  object
 8   actors                          17360 non-null  object
 9   original_release_date           16546 non-null  object
 10  streaming_release_date          17328 non-null  object
 11  runtime                         17398 non-null  float64
 12  production_company              17213 non-null  object
 13  tomatometer_status              17668 non-null  object
 14  tomatometer_rating              17668 non-null  float64
 15  tomatometer_count               17668 non-null  float64
 16  audience_status                 17264 non-null  object
 17  audience_rating                 17416 non-null  float64
 18  audience_count                  17415 non-null  float64
 19  tomatometer_top_critics_count   17712 non-null  int64
 20  tomatometer_fresh_critics_count 17712 non-null  int64
 21  tomatometer_rotten_critics_count 17712 non-null  int64
dtypes: float64(5), int64(3), object(14)
memory usage: 3.0+ MB
None
```

	rotten_tomatoes_link	movie_title	movie_info	critics_consensus	content_rating	
0	m/0814255	Percy Jackson & the Olympians: The Lightning T...	Always trouble-prone, the life of teenager Per...	Though it may seem like just another Harry Pot...	PG	A
1	m/0878835	Please Give	Kate (Catherine Keener) and her husband Alex (...	Nicole Holofcener's newest might seem slight i...	R	

	rotten_tomatoes_link	movie_title	movie_info	critics_consensus	content_rating	
2	m/10	10	A successful, middle-aged Hollywood songwriter...	Blake Edwards' bawdy comedy may not score a pe...	R	I
3	m/1000013-12_angry_men	12 Angry Men (Twelve Angry Men)	Following the closing arguments in a murder tr...	Sidney Lumet's feature debut is a superbly wri...	NR	
4	m/1000079-20000_leagues_under_the_sea	20,000 Leagues Under The Sea	In 1866, Professor Pierre M. Aronnax (Paul Luk...	One of Disney's finest live-action adventures,...	G	A

5 rows × 22 columns

```
Number of Null values in Rt_movies

rotten_tomatoes_link                 0
movie_title                          0
movie_info                         321
critics_consensus                 8578
content_rating                       0
genres                              19
directors                          194
authors                           1542
actors                             352
original_release_date             1166
streaming_release_date             384
runtime                            314
production_company                 499
tomatometer_status                  44
tomatometer_rating                  44
tomatometer_count                   44
audience_status                    448
audience_rating                    296
audience_count                     297
tomatometer_top_critics_count        0
tomatometer_fresh_critics_count      0
tomatometer_rotten_critics_count     0
dtype: int64

The Number of duplicated rows in rt_movies is 0
################################################################################
########
```

22 columns - not all of which will be required in this analysis, they may be useful for future work. Alot of this information is already been observed in IMDB dataframes, while it is useful to have multiple data sources in order to validate data accuracy and highlight discrepancies, that will not be conducted within the timeframe of this project. Instead this dataframe will be used for its unique data - reviews.

Cleaning Recommendations

- Drop 'critic_consensus' , 'movie_info' , 'tomatometer_top_critics_count' 'tomatometer_fresh_critics_count' 'tomatometer_rotten_critics_count' , 'streaming_release_date'
- Drop rows with null values in 'tomatometer_rating' , 'tomatometer_status', 'tomatometer_count'
- fill na of 'audience_status' , 'audience_rating' , 'audience_count' with median values

In [550]:

```
df_info(rt_reviews)
```

```
################################################################################
########

rt_reviews Database


The length of this dataframe is 1130017
Database shape (1130017, 8)


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1130017 entries, 0 to 1130016
Data columns (total 8 columns):
 #   Column                Non-Null Count    Dtype
---  ------                --------------    -----
 0   rotten_tomatoes_link  1130017 non-null  object
 1   critic_name           1111488 non-null  object
 2   top_critic            1130017 non-null  bool
 3   publisher_name        1130017 non-null  object
 4   review_type           1130017 non-null  object
 5   review_score          824081 non-null   object
 6   review_date           1130017 non-null  object
 7   review_content        1064211 non-null  object
dtypes: bool(1), object(7)
memory usage: 61.4+ MB
None
```

	rotten_tomatoes_link	critic_name	top_critic	publisher_name	review_type	review_score	rev
0	m/0814255	Andrew L. Urban	False	Urban Cinefile	Fresh	NaN	2(
1	m/0814255	Louise Keller	False	Urban Cinefile	Fresh	NaN	2(
2	m/0814255	NaN	False	FILMINK (Australia)	Fresh	NaN	2(
3	m/0814255	Ben McEachen	False	Sunday Mail (Australia)	Fresh	3.5/5	2(
4	m/0814255	Ethan Alter	True	Hollywood Reporter	Rotten	NaN	2(

```
Number of Null values in Rt_reviews

rotten_tomatoes_link         0
critic_name              18529
top_critic                   0
publisher_name               0
```

```
review_type                0
review_score          305936
review_date                0
review_content         65806
dtype: int64
The Number of duplicated rows in rt_reviews is 119471
################################################################################
########
```

This looks to be just individual review data, hence multiple entries for the same 'rotten_tomatoes_link'. 'rt_movies' dataframe already consolidates alot of review data that will be useful for this project. Further work could add granularity to the link between review ratings and profit or ROI by analysing any potential link between specific reviewers scores and a movies' success. For this project, this dataframe will not be used.

In [551]:

```
df_info(tmdb)
```

```
################################################################################
########

tmdb Database


The length of this dataframe is 26517
Database shape (26517, 9)


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26517 entries, 0 to 26516
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   genre_ids          26517 non-null  object 
 1   id                 26517 non-null  int64  
 2   original_language  26517 non-null  object 
 3   original_title     26517 non-null  object 
 4   popularity         26517 non-null  float64
 5   release_date       26517 non-null  object 
 6   title              26517 non-null  object 
 7   vote_average       26517 non-null  float64
 8   vote_count         26517 non-null  int64  
dtypes: float64(2), int64(2), object(5)
memory usage: 1.8+ MB
None
```

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_av
0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	
1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	
2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	
3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	
4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	

```
Number of Null values in Tmdb

genre_ids            0
id                   0
original_language    0
original_title       0
popularity           0
release_date         0
title                0
```

```
vote_average         0
vote_count           0
dtype: int64

The Number of duplicated rows in tmdb is 1020
################################################################################
########
```

This dataframe could be valuable to cross check conclusions made using other dataframes as it contains similar information but from yet another source.

Cleaning Recommendations

- drop 'genre_ids', 'id', 'original_language', 'original_title'

In [552]:

```
df_info(budgets)
```

```
################################################################################
########

budgets Database


The length of this dataframe is 5782
Database shape (5782, 6)


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5782 entries, 0 to 5781
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   id                 5782 non-null   int64 
 1   release_date       5782 non-null   object
 2   movie              5782 non-null   object
 3   production_budget  5782 non-null   object
 4   domestic_gross     5782 non-null   object
 5   worldwide_gross    5782 non-null   object
dtypes: int64(1), object(5)
memory usage: 271.2+ KB
None
```

	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

```
Number of Null values in Budgets

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

The Number of duplicated rows in budgets is 0
################################################################################
########
```

Valuable dataframe, although short at 5782 entries. This will be used to infer patterns between profit / ROI / budgets.

Cleaning Recommendations

- drop 'id' column
- convert 'production_budget', 'domestic_gross', 'worldwide_gross' to float - remove commas and dollar signs.
- convert 'release_date' to datetime - extract month and year and form separate columns for both then remove 'release_date'

In [553]:

```
df_info(gross)
```

```
################################################################################
########

gross Database


The length of this dataframe is 3387
Database shape (3387, 5)


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3387 entries, 0 to 3386
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   title           3387 non-null   object 
 1   studio          3382 non-null   object 
 2   domestic_gross  3359 non-null   float64
 3   foreign_gross   2037 non-null   object 
 4   year            3387 non-null   int64  
dtypes: float64(1), int64(1), object(3)
memory usage: 132.4+ KB
None
```

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010

```
Number of Null values in Gross

title               0
studio              5
domestic_gross     28
foreign_gross    1350
year                0
dtype: int64

The Number of duplicated rows in gross is 0
################################################################################
########
```

Dataframe offers similar information as 'budgets' dataframe however is shorter. It does add studio information which could prove useful when evaluating what the competition is doing.

Data Preparation

The following dataframes will form the bulk of this project:

- budgets
- imdb_name, imdb_basics, imdb_ratings, imdb_principals
- rt_movies

Budgets

Starting with 'budgets', first lets drop the 'id' column

In [554]:

```
budgets.drop('id', axis=1, inplace=True)
```

Create a function for cleaning up columns that should be a number - remove dollar signs and commas and convert to float.

In [555]:

```
def convert_col_float(df, col):
    '''takes in a dataframe and a column name within the dataframe, removes commas and doll
    df[col] = df[col].str.replace(',','')
    df[col] = df[col].str.replace('$','')
    df[col] = df[col].astype(float)
    return df[col]
```

In [556]:

```
convert_col_float(budgets, 'production_budget')
convert_col_float(budgets, 'domestic_gross')
convert_col_float(budgets, 'worldwide_gross')
budgets.head()
```

Out[556]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	Dec 18, 2009	Avatar	425000000.0	760507625.0	2.776345e+09
1	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09
2	Jun 7, 2019	Dark Phoenix	350000000.0	42762350.0	1.497624e+08
3	May 1, 2015	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09
4	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000.0	620181382.0	1.316722e+09

Let's now convert the 'release_date' column to something useable.

In [557]:

```
budgets['release_date'] = pd.to_datetime(budgets['release_date'])
budgets.head()
```

Out[557]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	2009-12-18	Avatar	425000000.0	760507625.0	2.776345e+09
1	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09
2	2019-06-07	Dark Phoenix	350000000.0	42762350.0	1.497624e+08
3	2015-05-01	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09
4	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000.0	620181382.0	1.316722e+09

I can foresee both year and month data being a useful parameter for group, create a column for both by extracting the information from 'release_date'

In [558]:

```
budgets['month'] = budgets['release_date'].dt.month
budgets['year'] = budgets['release_date'].dt.year
budgets.head()
```

Out[558]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	month	year
0	2009-12-18	Avatar	425000000.0	760507625.0	2.776345e+09	12	2009
1	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09	5	2011
2	2019-06-07	Dark Phoenix	350000000.0	42762350.0	1.497624e+08	6	2019
3	2015-05-01	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09	5	2015
4	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000.0	620181382.0	1.316722e+09	12	2017

I will be using this dataframe to analyse the relationship between budgets and profit, in order to do this I will need to create a profit column. A quick google tells me that in 'The Numbers' database, the 'worldwide_gross' is inclusive of the 'domestic_gross'. The profit will therefore simply be the worldwide_gross - production_budget. This will be easier to view when the profit is in $ million.

The return on investment (roi) column is the ratio between budget and profit, in this instance it will be the profit column divided the budget column i.e for every dollar you invest, how many dollars profit will you receive?

In [559]:

```
#convert 'production_budget' , 'domestic_gross' , 'worldwide_gross' to millions units.

budgets['production_budget'] = budgets['production_budget'] / 1000000
budgets['domestic_gross'] = budgets['domestic_gross'] / 1000000
budgets['worldwide_gross'] = budgets['worldwide_gross'] / 1000000

# commented out this cell as to not accidentally run again
```

In [560]:

```
#creating profit and roi columns
budgets['profit'] = budgets['worldwide_gross'] - budgets['production_budget']
budgets['roi'] = budgets['profit'] / budgets['production_budget']
```

In [561]:

```
budgets.head()
```

Out[561]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	month	year
0	2009-12-18	Avatar	425.0	760.507625	2776.345279	12	2009
1	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410.6	241.063875	1045.663875	5	2011
2	2019-06-07	Dark Phoenix	350.0	42.762350	149.762350	6	2019
3	2015-05-01	Avengers: Age of Ultron	330.6	459.005868	1403.013963	5	2015
4	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317.0	620.181382	1316.721747	12	2017

Using the .describe() method, check the distribution of values in each column, this is useful for flagging up potential issues

In [562]:

```
budgets.describe()
```

Out[562]:

	production_budget	domestic_gross	worldwide_gross	month	year	I
count	5782.000000	5782.000000	5782.000000	5782.000000	5782.000000	5782.00
mean	31.587757	41.873327	91.487461	7.050675	2003.967139	59.89
std	41.812077	68.240597	174.719969	3.480147	12.724386	146.08
min	0.001100	0.000000	0.000000	1.000000	1915.000000	-200.23
25%	5.000000	1.429534	4.125415	4.000000	2000.000000	-2.18
50%	17.000000	17.225945	27.984448	7.000000	2007.000000	8.55
75%	40.000000	52.348662	97.645837	10.000000	2012.000000	60.96
max	425.000000	936.662225	2776.345279	12.000000	2020.000000	2351.34

The first thing that stands out here is the minimum values in both the domestic and worldwide gross columns. There may be a reason for why a movie would have not earned any money as of yet - the main one being it might not have been released yet. Either way, a movie with zero domestic gross and worldwide gross is extremely unlikely and likely erroneous and I will remove them from this analysis.

In [563]:

```
# check how many rows are impacted

len(budgets[(budgets['domestic_gross']==0) & (budgets['worldwide_gross']==0)])
```

Out[563]:

```
367
```

In [564]:

```
# remove the rows from the database

budgets = budgets[(budgets['domestic_gross']!=0) & (budgets['worldwide_gross']!=0)]
```

In [565]:

```
# splitting the production budget into categories to explore patterns

bins = [0, 10, 25, 50, 100, 200, 300, np.inf]
names = ['<10m', '10-25m', '25-50m', '50-100m', '100-200m', '200-300m', '>300m']

budgets['budget_bin'] = pd.cut(budgets['production_budget'], bins, labels=names)

budgets['p&l'] = budgets['roi'].map(lambda x : 1 if x >= 0 else 0)
budgets.head()
```

Out[565]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	month	year
0	2009-12-18	Avatar	425.0	760.507625	2776.345279	12	2009
1	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410.6	241.063875	1045.663875	5	2011
2	2019-06-07	Dark Phoenix	350.0	42.762350	149.762350	6	2019
3	2015-05-01	Avengers: Age of Ultron	330.6	459.005868	1403.013963	5	2015
4	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317.0	620.181382	1316.721747	12	2017

The budgets dataframe is now ready for some preliminary analysis.

rt_movies

- Drop 'critic_consensus' , 'movie_info' , 'tomatometer_top_critics_count' 'tomatometer_fresh_critics_count' 'tomatometer_rotten_critics_count' , 'streaming_release_date'
- Drop rows with null values in 'tomatometer_rating' , 'tomatometer_status', 'tomatometer_count'
- fill na of 'audience_status' , 'audience_rating' , 'audience_count' with median values

In [566]:

```
# as per bullet point one, dropping the neccessary columns

cols_to_remove= ['critics_consensus' , 'movie_info' , 'tomatometer_top_critics_count',\
 'tomatometer_fresh_critics_count', 'tomatometer_rotten_critics_count',\
 'streaming_release_date']

rt_movies.drop(cols_to_remove, axis=1, inplace=True)
```

In [567]:

```
rt_movies.head()
```

Out[567]:

	rotten_tomatoes_link	movie_title	content_rating	genres	directors	authors	
0	m/0814255	Percy Jackson & the Olympians: The Lightning T...	PG	Action & Adventure, Comedy, Drama, Science Fic...	Chris Columbus	Craig Titley, Chris Columbus, Rick Riordan	A
1	m/0878835	Please Give	R	Comedy	Nicole Holofcener	Nicole Holofcener	
2	m/10	10	R	Comedy, Romance	Blake Edwards	Blake Edwards	B
3	m/1000013-12_angry_men	12 Angry Men (Twelve Angry Men)	NR	Classics, Drama	Sidney Lumet	Reginald Rose	
4	m/1000079-20000_leagues_under_the_sea	20,000 Leagues Under The Sea	G	Action & Adventure, Drama, Kids & Family	Richard Fleischer	Earl Felton	

In [568]:

```
# removing rows that have null values in the following 3 columns

rt_movies.dropna(subset=['tomatometer_rating' , 'tomatometer_status', 'tomatometer_count'],
```

In [569]:

```
df_info(rt_movies)
```

```
################################################################################
########

rt_movies Database


The length of this dataframe is 17668
Database shape (17668, 16)


<class 'pandas.core.frame.DataFrame'>
Int64Index: 17668 entries, 0 to 17711
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   rotten_tomatoes_link   17668 non-null  object
 1   movie_title            17668 non-null  object
 2   content_rating         17668 non-null  object
 3   genres                 17649 non-null  object
 4   directors              17475 non-null  object
 5   authors                16134 non-null  object
 6   actors                 17316 non-null  object
 7   original_release_date  16514 non-null  object
 8   runtime                17384 non-null  float64
 9   production_company     17175 non-null  object
 10  tomatometer_status     17668 non-null  object
 11  tomatometer_rating     17668 non-null  float64
 12  tomatometer_count      17668 non-null  float64
 13  audience_status        17255 non-null  object
 14  audience_rating        17407 non-null  float64
 15  audience_count         17406 non-null  float64
dtypes: float64(5), object(11)
memory usage: 2.3+ MB
None
```

	rotten_tomatoes_link	movie_title	content_rating	genres	directors	authors
0	m/0814255	Percy Jackson & the Olympians: The Lightning T...	PG	Action & Adventure, Comedy, Drama, Science Fic...	Chris Columbus	Craig Titley, Chris Columbus, Rick Riordan
1	m/0878835	Please Give	R	Comedy	Nicole Holofcener	Nicole Holofcener
2	m/10	10	R	Comedy, Romance	Blake Edwards	Blake Edwards

	rotten_tomatoes_link	movie_title	content_rating	genres	directors	authors
3	m/1000013-12_angry_men	12 Angry Men (Twelve Angry Men)	NR	Classics, Drama	Sidney Lumet	Reginald Rose
4	m/1000079-20000_leagues_under_the_sea	20,000 Leagues Under The Sea	G	Action & Adventure, Drama, Kids & Family	Richard Fleischer	Earl Felton

```
Number of Null values in Rt_movies

rotten_tomatoes_link        0
movie_title                 0
content_rating              0
genres                     19
directors                 193
authors                  1534
actors                    352
original_release_date    1154
runtime                   284
production_company        493
tomatometer_status          0
tomatometer_rating          0
tomatometer_count           0
audience_status           413
audience_rating           261
audience_count            262
dtype: int64

The Number of duplicated rows in rt_movies is 0
################################################################################
########
```

In [570]:

```
# create a function for filling null values in columns with their median values

def median_fill(df, col):
    df[col] = df[col].fillna((df[col].median()))
```

In [571]:

```
#filling columns with the median value of the column
median_fill(rt_movies, 'audience_rating')
median_fill(rt_movies, 'audience_count')
```

In [572]:

```
# replacing the audience status column with the conditions taken from Rotten Tomatoes Websi
conditions = [(rt_movies['audience_rating'] < 60), (rt_movies['audience_rating'] >= 60)]

# create a list of the values we want to assign for each condition
values = ['Spilled', 'Upright']

# create a new column and use np.select to assign values to it using our lists as arguments
rt_movies['audience_rating_new'] = np.select(conditions, values)

# dropping original column with missing values
rt_movies.drop(['audience_status'], axis=1, inplace=True)
```

In [573]:

```
rt_movies.head()
```

Out[573]:

	rotten_tomatoes_link	movie_title	content_rating	genres	directors	authors	
0	m/0814255	Percy Jackson & the Olympians: The Lightning T...	PG	Action & Adventure, Comedy, Drama, Science Fic...	Chris Columbus	Craig Titley, Chris Columbus, Rick Riordan	A
1	m/0878835	Please Give	R	Comedy	Nicole Holofcener	Nicole Holofcener	
2	m/10	10	R	Comedy, Romance	Blake Edwards	Blake Edwards	B
3	m/1000013-12_angry_men	12 Angry Men (Twelve Angry Men)	NR	Classics, Drama	Sidney Lumet	Reginald Rose	
4	m/1000079-20000_leagues_under_the_sea	20,000 Leagues Under The Sea	G	Action & Adventure, Drama, Kids & Family	Richard Fleischer	Earl Felton	

Although there are still some imperfections in this dataset, its use will be primarily for checking review ratings vs profit and roi, and the review columns and movie_title columns are complete for now.

imdb databases

imdb_name

- Drop 'birth_year' and 'death_year' columns
- 'primary_profession' appears to contain the same information as 'category' in 'principals' dataframe. This could be dropped.
- merge with principals dataframe

In [574]:

```
# dropping 'birth_year' , 'death_year' & 'primary_profession' columns

imdb_name.drop(['birth_year', 'death_year', 'primary_profession'], axis=1, inplace=True)
```

Now lets merge this dataframe with the prinicpals dataframe, its worth reminding ourselves the structure of both.

In [575]:

```
df_info(imdb_name)
```

```
################################################################################
########

imdb_name Database


The length of this dataframe is 606648
Database shape (606648, 3)


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 606648 entries, 0 to 606647
Data columns (total 3 columns):
 #   Column           Non-Null Count   Dtype 
---  ------           --------------   ----- 
 0   nconst           606648 non-null  object
 1   primary_name     606648 non-null  object
 2   known_for_titles  576444 non-null  object
dtypes: object(3)
memory usage: 13.9+ MB
None
```

	nconst	primary_name	known_for_titles
0	nm0061671	Mary Ellen Bauder	tt0837562,tt2398241,tt0844471,tt0118553
1	nm0061865	Joseph Bauer	tt0896534,tt6791238,tt0287072,tt1682940
2	nm0062070	Bruce Baum	tt1470654,tt0363631,tt0104030,tt0102898
3	nm0062195	Axel Baumann	tt0114371,tt2004304,tt1618448,tt1224387
4	nm0062798	Pete Baxter	tt0452644,tt0452692,tt3458030,tt2178256

```
Number of Null values in Imdb_name

nconst                 0
primary_name           0
known_for_titles    30204
dtype: int64

The Number of duplicated rows in imdb_name is 0
################################################################################
########
```

In [576]:

```
df_info(principals)
```

```
################################################################################
########

principals Database


The length of this dataframe is 1028186
Database shape (1028186, 6)


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1028186 entries, 0 to 1028185
Data columns (total 6 columns):
 #   Column      Non-Null Count    Dtype 
---  ------      --------------    ----- 
 0   tconst      1028186 non-null  object
 1   ordering    1028186 non-null  int64 
 2   nconst      1028186 non-null  object
 3   category    1028186 non-null  object
 4   job         177684 non-null   object
 5   characters  393360 non-null   object
dtypes: int64(1), object(5)
memory usage: 47.1+ MB
None
```

	tconst	ordering	nconst	category	job	characters
0	tt0111414	1	nm0246005	actor	NaN	["The Man"]
1	tt0111414	2	nm0398271	director	NaN	NaN
2	tt0111414	3	nm3739909	producer	producer	NaN
3	tt0323808	10	nm0059247	editor	NaN	NaN
4	tt0323808	1	nm3579312	actress	NaN	["Beth Boothby"]

```
Number of Null values in Principals

tconst             0
ordering           0
nconst             0
category           0
job           850502
characters    634826
dtype: int64

The Number of duplicated rows in principals is 0
################################################################################
########
```

In [577]:

```
# merge the principals dataframe with the imdb_name dataframe using 'nconst'

principals_names = principals.merge(imdb_name, how='left', on='nconst')
df_info(principals_names)
```

```
################################################################################
########

principals_names Database


The length of this dataframe is 1028186
Database shape (1028186, 8)


<class 'pandas.core.frame.DataFrame'>
Int64Index: 1028186 entries, 0 to 1028185
Data columns (total 8 columns):
 #   Column            Non-Null Count    Dtype
---  ------            --------------    -----
 0   tconst            1028186 non-null  object
 1   ordering          1028186 non-null  int64
 2   nconst            1028186 non-null  object
 3   category          1028186 non-null  object
 4   job               177684 non-null   object
 5   characters        393360 non-null   object
 6   primary_name      1027912 non-null  object
 7   known_for_titles  997445 non-null   object
dtypes: int64(1), object(7)
memory usage: 70.6+ MB
None
```

	tconst	ordering	nconst	category	job	characters	primary_name	
0	tt0111414	1	nm0246005	actor	NaN	["The Man"]	Tommy Dysart	tt0093120,tt00
1	tt0111414	2	nm0398271	director	NaN	NaN	Frank Howson	tt0104271,tt00
2	tt0111414	3	nm3739909	producer	producer	NaN	Barry Porter-Robinson	tt0290884,tt01
3	tt0323808	10	nm0059247	editor	NaN	NaN	Sean Barton	tt0402910,tt10
4	tt0323808	1	nm3579312	actress	NaN	["Beth Boothby"]	Brittania Nicol	

```
Number of Null values in Principals_names

tconst                  0
ordering                0
nconst                  0
category                0
job                850502
characters         634826
primary_name          274
known_for_titles    30741
dtype: int64

The Number of duplicated rows in principals_names is 0
################################################################################
```

########

As discussed in the previous sections, ordering, job and characters columns can be dropped, known_for_titles may prove useful, it will be kept for now.

In [578]:

```
# drop columns as discussed
principals_names.drop(['job', 'characters', 'ordering'], axis=1, inplace=True)
principals_names.head()
```

Out[578]:

	tconst	nconst	category	primary_name	known_for_titles
0	tt0111414	nm0246005	actor	Tommy Dysart	tt0093120,tt0076974,tt0084296,tt0077064
1	tt0111414	nm0398271	director	Frank Howson	tt0104271,tt0094789,tt0102076,tt0111414
2	tt0111414	nm3739909	producer	Barry Porter-Robinson	tt0290884,tt0101374,tt0111414,tt1566940
3	tt0323808	nm0059247	editor	Sean Barton	tt0402910,tt1022883,tt0086190,tt0490181
4	tt0323808	nm3579312	actress	Brittania Nicol	tt0323808

In [579]:

```
principals_names['category'].value_counts()
```

Out[579]:

```
actor                  256718
director               146393
actress                146208
producer               113724
cinematographer         80091
composer                77063
writer                  74357
self                    65424
editor                  55512
production_designer      9373
archive_footage          3307
archive_sound              16
Name: category, dtype: int64
```

This dataframe will prove useful when looking for the best talent to approach for this project

imdb_basics

- drop the 'original_title' columns.
- merge with the imdb_ratings dataframe
- Split the genres column

In [580]:

```
df_info(imdb_basics)
```

```
################################################################################
########

imdb_basics Database


The length of this dataframe is 146144
Database shape (146144, 6)


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   tconst           146144 non-null  object
 1   primary_title    146144 non-null  object
 2   original_title   146123 non-null  object
 3   start_year       146144 non-null  int64
 4   runtime_minutes  114405 non-null  float64
 5   genres           140736 non-null  object
dtypes: float64(1), int64(1), object(4)
memory usage: 6.7+ MB
None
```

	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

```
Number of Null values in Imdb_basics

tconst                0
primary_title         0
original_title       21
start_year            0
runtime_minutes    31739
genres              5408
dtype: int64

The Number of duplicated rows in imdb_basics is 0
################################################################################
```

##########

In [581]:

```
# drop original title column
imdb_basics.drop(['original_title'], axis = 1, inplace = True)
```

In [582]:

```
#merge with ratings
imdb_basics = imdb_basics.merge(imdb_ratings, how='left', on='tconst')
```

In [583]:

```
imdb_basics.head()
```

Out[583]:

	tconst	primary_title	start_year	runtime_minutes	genres	averagerating	nu
0	tt0063540	Sunghursh	2013	175.0	Action,Crime,Drama	7.0	
1	tt0066787	One Day Before the Rainy Season	2019	114.0	Biography,Drama	7.2	
2	tt0069049	The Other Side of the Wind	2018	122.0	Drama	6.9	
3	tt0069204	Sabse Bada Sukh	2018	NaN	Comedy,Drama	6.1	
4	tt0100275	The Wandering Soap Opera	2017	80.0	Comedy,Drama,Fantasy	6.5	

In [584]:

```
df_info(imdb_basics)
```

```
################################################################################
########

imdb_basics Database


The length of this dataframe is 146144
Database shape (146144, 7)


<class 'pandas.core.frame.DataFrame'>
Int64Index: 146144 entries, 0 to 146143
Data columns (total 7 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   tconst           146144 non-null  object
 1   primary_title    146144 non-null  object
 2   start_year       146144 non-null  int64
 3   runtime_minutes  114405 non-null  float64
 4   genres           140736 non-null  object
 5   averagerating    73856 non-null   float64
 6   numvotes         73856 non-null   float64
dtypes: float64(3), int64(1), object(3)
memory usage: 8.9+ MB
None
```

	tconst	primary_title	start_year	runtime_minutes	genres	averagerating	nu
0	tt0063540	Sunghursh	2013	175.0	Action,Crime,Drama	7.0	
1	tt0066787	One Day Before the Rainy Season	2019	114.0	Biography,Drama	7.2	
2	tt0069049	The Other Side of the Wind	2018	122.0	Drama	6.9	
3	tt0069204	Sabse Bada Sukh	2018	NaN	Comedy,Drama	6.1	
4	tt0100275	The Wandering Soap Opera	2017	80.0	Comedy,Drama,Fantasy	6.5	

```
Number of Null values in Imdb_basics

tconst                0
primary_title         0
start_year            0
runtime_minutes    31739
genres              5408
averagerating      72288
numvotes           72288
dtype: int64

The Number of duplicated rows in imdb_basics is 0
```

```
##########################################################################
##########
```

In [585]:

```
imdb_basics.describe()
```

Out[585]:

	start_year	runtime_minutes	averagerating	numvotes
count	146144.000000	114405.000000	73856.000000	7.385600e+04
mean	2014.621798	86.187247	6.332729	3.523662e+03
std	2.733583	166.360590	1.474978	3.029402e+04
min	2010.000000	1.000000	1.000000	5.000000e+00
25%	2012.000000	70.000000	5.500000	1.400000e+01
50%	2015.000000	87.000000	6.500000	4.900000e+01
75%	2017.000000	99.000000	7.400000	2.820000e+02
max	2115.000000	51420.000000	10.000000	1.841066e+06

Genres dataframe

It will be useful to analyse which genres are the most profitable or have the highest ROI. To do this I will need to combine the 'budgets' dataframe with another that contains information on the movie genre.

In [586]:

```
# dropna from genre column in rt_movies
rt_movies.dropna(subset=['genres'], inplace=True)

### Create dataframe that stacks movies by genres using rt_movies dataframe
genres_df = rt_movies.set_index('movie_title').genres.str.split(',', expand=True).stack().r
genres_df.columns=['primary_title', 'genre']

# rt_movies contains no profit or budget information, requires merge with 'budgets'
genres_df = genres_df.merge(budgets, how='left', left_on='primary_title', right_on='movie')
```

Profit and ROI are key for this analysis, therefore there is no value in imputing such a large number of missing values, insights wouldn't be accurate. Therefore I will remove rows that have null values in what were the 'budgets' dataframe columns

In [587]:

```
### drop null values from legacy 'budgets' dataframe that are now null values following the
genres_df.dropna(subset=['release_date'], inplace=True)

### clean genres column and create list of genres
genres_df['genre'] = genres_df['genre'].str.strip().str.replace('&', 'and')
list_genres = list(genres_df.genre.unique())

## create list of genres split up for wordcloud
genres = []
for i in list_genres:
    for entry in i.split(' '):
        if entry != 'and':
            #print(entry)
            genres.append(entry)
        else:
            continue

#import the prerequisites to generate word cloud
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

plt.figure(figsize=[15,15])
# join the list and lowercase all the words
text = ' '.join(genres).lower()

#create the wordcloud object
wordcloud = WordCloud(collocations=True, background_color='white').generate(text)

#wordcloud.to_file("genres.png");
#plot the wordcloud object
#plt.imshow(wordcloud, interpolation='bilInear')
#plt.axis('off')
#plt.show();


# new dataframe grouping mean values by genre sorted on profit
new_genre_df = genres_df[genres_df['year']>2009].groupby(['genre']).mean().sort_values('pro
new_genre_df = new_genre_df.reset_index()



# # same dataframe sorted by p&l
genres_df.dropna(subset=['p&l'], inplace=True)
profit_loss = genres_df[genres_df['year']>2009].groupby(['genre']).mean().sort_values('p&l'

# # merge genres dataframe with rt_movies to explore ratings vs profits
ratings_profits = genres_df.merge(rt_movies, how='left', left_on='movie', right_on='movie_t

# #create animation dataframe
animation_df = rt_movies[rt_movies['genres'].str.contains('Animation')]

# dataframe for animation sorted by profits
animation_by_profit = ratings_profits[ratings_profits['genre'].str.contains('Animation')].s
```

```
<Figure size 1080x1080 with 0 Axes>
```

In [588]:

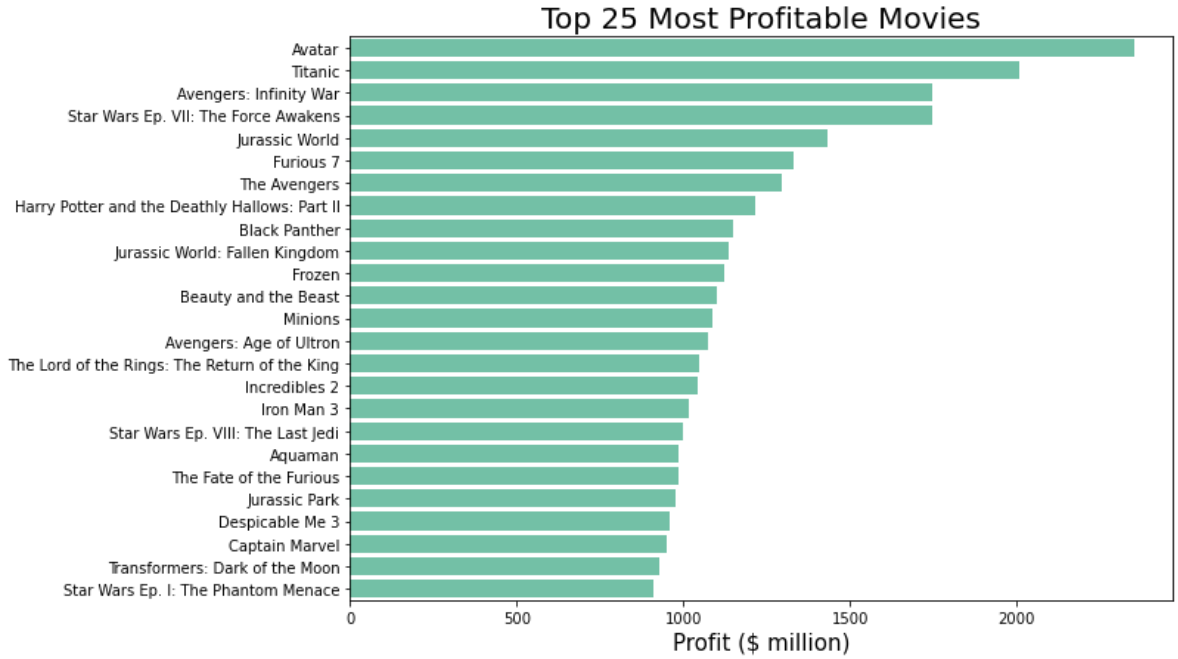
```
## count strings in column

def splitter_counter(df, col):
    ''' takes in a dataframe and column name and returns a dictionary\
    with key:value is string : number of times string appears in column'''
    list_strings = []
    for entry in df[col].str.split(','):
        for i in entry:
            list_strings.append(i)
    strings = [x.strip(' ') for x in list_strings]
    for i in strings:
        if len(i) < 4:
            strings.remove(i)
    #print('Jr.' in strings)
    #print(len(strings))
    strings_dict = {}
    for entry in strings:
        if entry not in strings_dict:
            strings_dict[entry] = len(df[df[col].str.contains(entry)])
        else:
            continue
    #print(len(strings_dict))
    return sorted(strings_dict.items(), key = lambda item: item[1], reverse = True)
```

Data Visualizations

What are the most successful movies of all time?

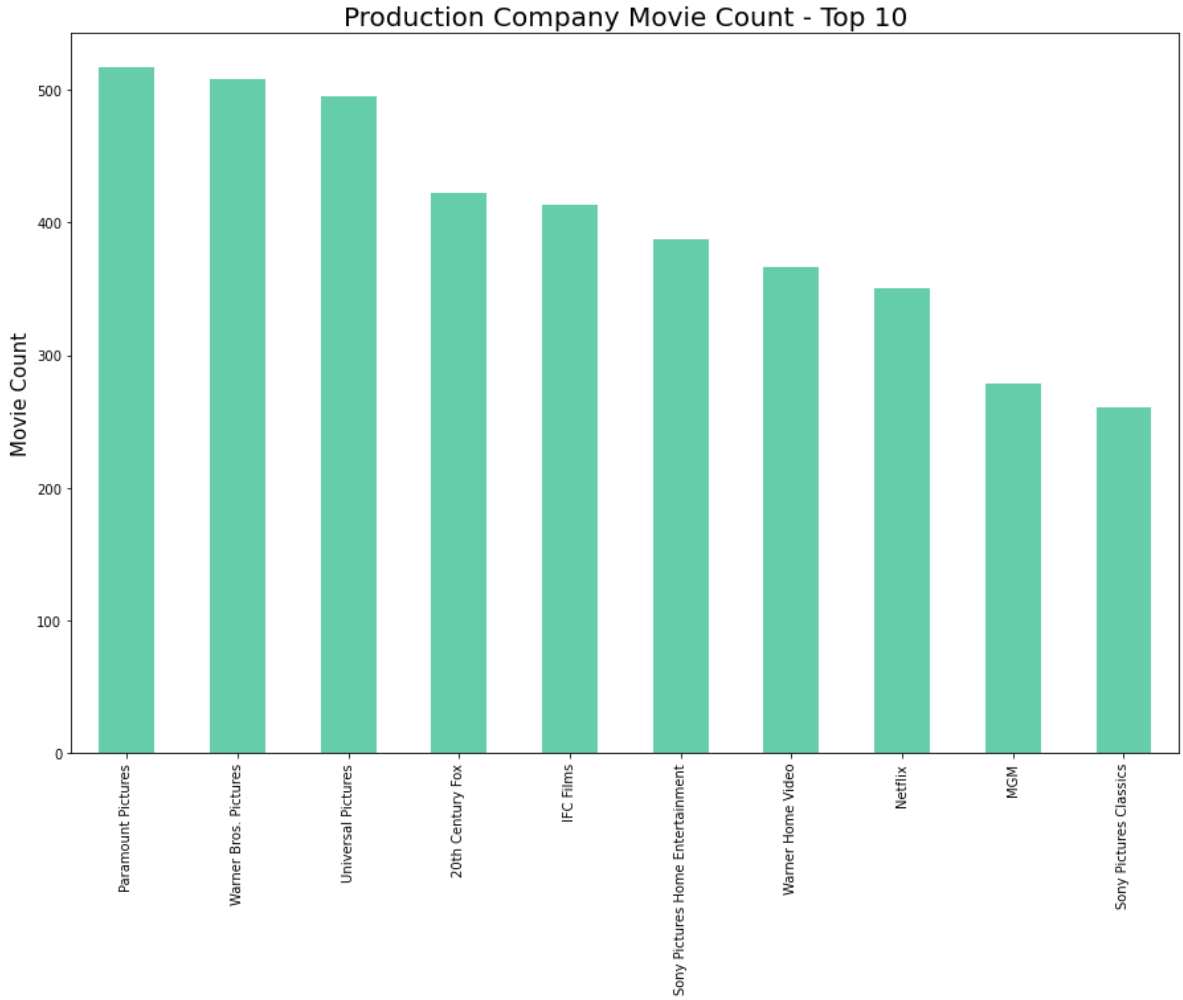
In [589]:

```
plt.figure(figsize = (10,7))
figure_1 = sns.barplot(y = 'movie', x = 'profit', data=budgets.sort_values('profit',ascendi
                       color = 'mediumaquamarine');
figure_1.set_title('Top 25 Most Profitable Movies', fontsize=20);
figure_1.set_ylabel('', fontsize=15)
figure_1.set_xlabel('Profit ($ million)', fontsize=15);
```



In [590]:

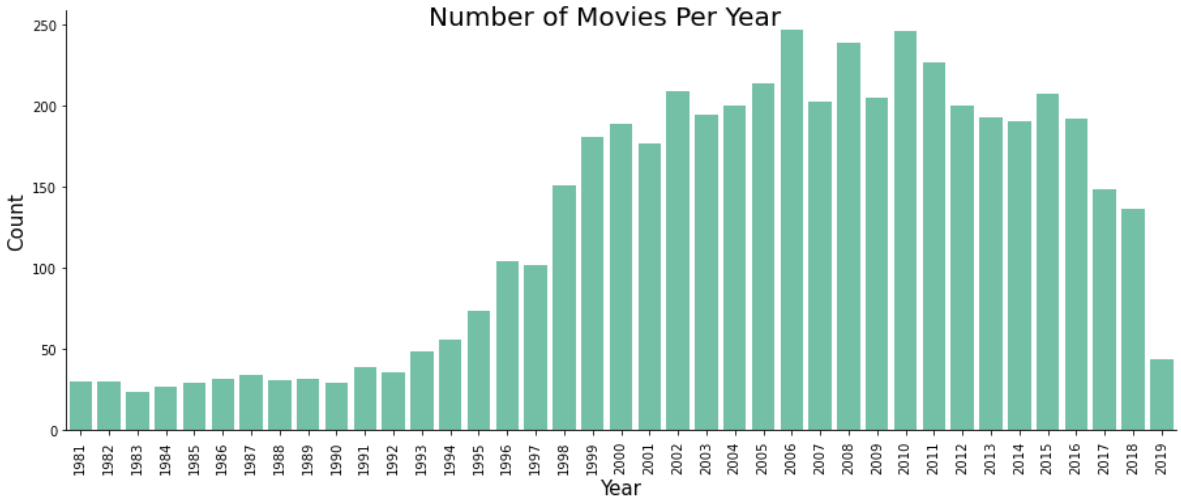
```
plt.figure(figsize=(15,10))
figure_13 = rt_movies.production_company.value_counts()[0:10].plot(kind = 'bar', color = 'm
figure_13.set_ylabel('Movie Count', fontsize=15)
figure_13.set_title('Production Company Movie Count - Top 10', fontsize=20);
figure_13.figure.savefig('./images/top10companies.png', bbox_inches='tight')
```



Is the competition becoming more fierce? Let's explore the number of movies being made through time

In [591]:

```
#plt.figure(figsize = (10,7));
figure_2 = sns.catplot(x = 'year', kind = 'count', data=budgets[budgets['year']>1980],\
                       color = 'mediumaquamarine', aspect = 2.5);
figure_2.set_xticklabels(rotation=90);
figure_2.set_xlabels('Year', fontsize=15);
figure_2.set_ylabels('Count', fontsize=15);
figure_2.fig.suptitle('Number of Movies Per Year', fontsize=20);
```



Looking at the 'budgets' dataframe it seems more movies are being made now than ever.Although this may not be a perfect dataset, it is likely that this plot does reflect the truth to some extent. This trend can be validated by cross checking against another database.

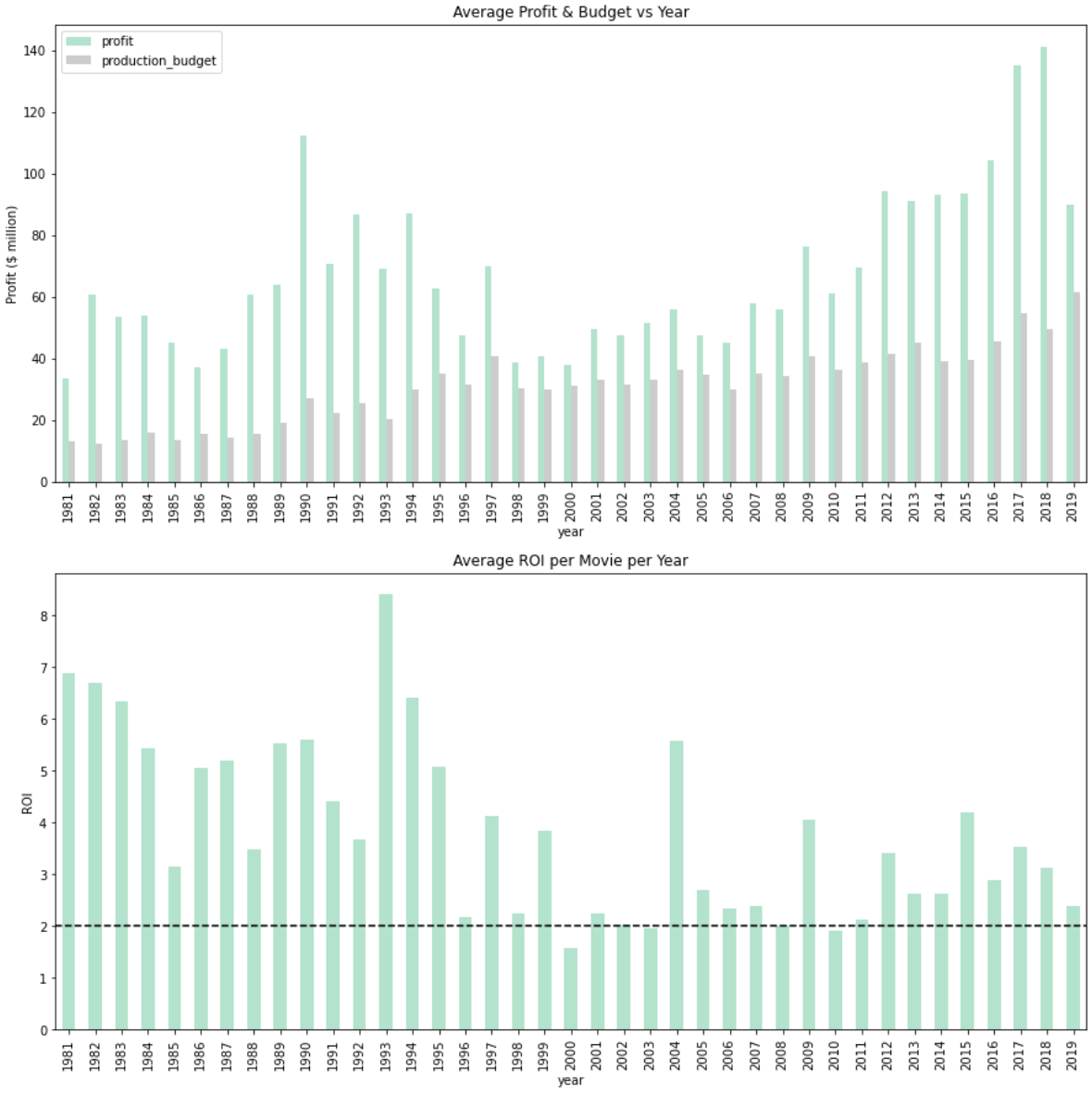
The business of making movies is more competitive than ever before, this is not considering streaming which is a different type of competitor - one that is keeping audiences away from theatres altogether!

Let's talk about Money

Everyone wanting to make movies, must mean the profits are great, right?

In [592]:

```
figure_3, axes = plt.subplots(2,1)

budgets[budgets['year']>1980].groupby(budgets['year'])\
        [['profit', 'production_budget']].mean().plot.bar(figsize=(15,15), ax = axes[0], co
budgets[budgets['year']>1980].groupby(budgets['year'])['roi'].mean().plot.bar(colormap='Pas
axes[0].set_title('Average Profit & Budget vs Year')
axes[0].set_ylabel('Profit ($ million)')
axes[1].set_title('Average ROI per Movie per Year')
axes[1].set_ylabel('ROI');
axes[1].axline(xy1= (1981, 2), slope=0, color='black', linestyle='--');
```



So yes, profits are increasing with time in absolute terms but so are the production budgets. This means when you compare Return on Investment (ROI) through the years it has actually decreased compared to the hey days of the 1990s.

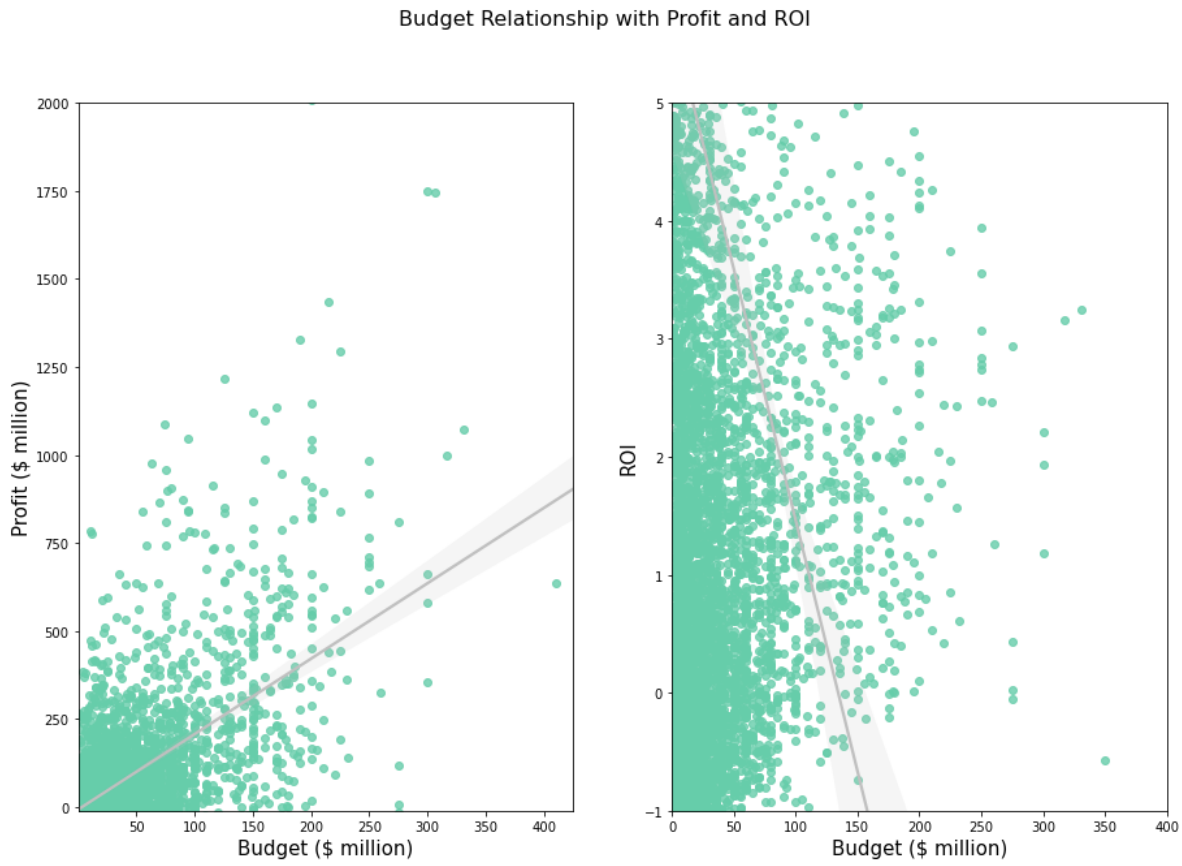
Making movies however is still a profitable business, with an ROI of between 2-3 being the norm in recent years.

Budget vs Profit vs ROI

How much can Microsoft expect to spend per movie? The more you spend the more you make?

In [593]:

```
figure_4, axes = plt.subplots(1,2, figsize=(15,10))
figure_4.suptitle('Budget Relationship with Profit and ROI', fontsize=16)
axes[0] = sns.regplot(data=budgets, x='production_budget', y='profit', scatter_kws={'color'
axes[0].set_ylim(-10,2000)
axes[1] = sns.regplot(data=budgets, x='production_budget', y='roi', scatter_kws={'color': '
axes[1].set_xlim(0,400)
axes[1].set_ylim(-1,5)
axes[1].set_ylabel('ROI', fontsize=15)
axes[0].set_ylabel('Profit ($ million)', fontsize=15)
axes[0].set_xlabel('Budget ($ million)', fontsize=15)
axes[1].set_xlabel('Budget ($ million)', fontsize=15);
```



In [594]:

```
budgets.corr()['production_budget']
```

Out[594]:

```
production_budget    1.000000
domestic_gross       0.678207
worldwide_gross      0.744875
month                0.042816
year                 0.217862
profit               0.605544
roi                 -0.058977
p&l                  0.165525
Name: production_budget, dtype: float64
```

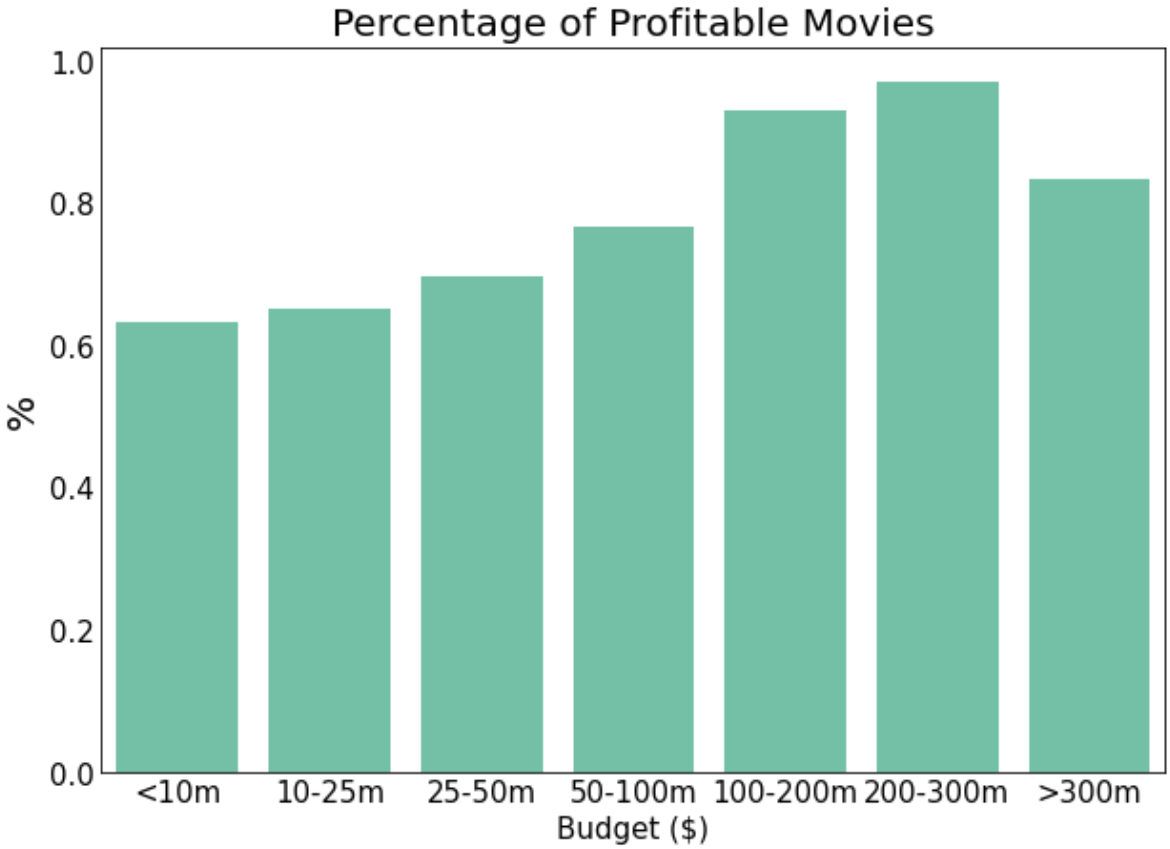
- The plots and accompanying correlation coeffients tell a different story for profit and ROI.
- There is a moderate positive correlation between production budget and profit (0.61)
- There is practically no correlation between profit and ROI (slightly negative (-0.06))

Spending more on production budget itself does not guarantee profit, however, if a movie is profitable, the profits are more likely to be higher the higher the production budget is.

More analysis is required before a production budget could be recommended.

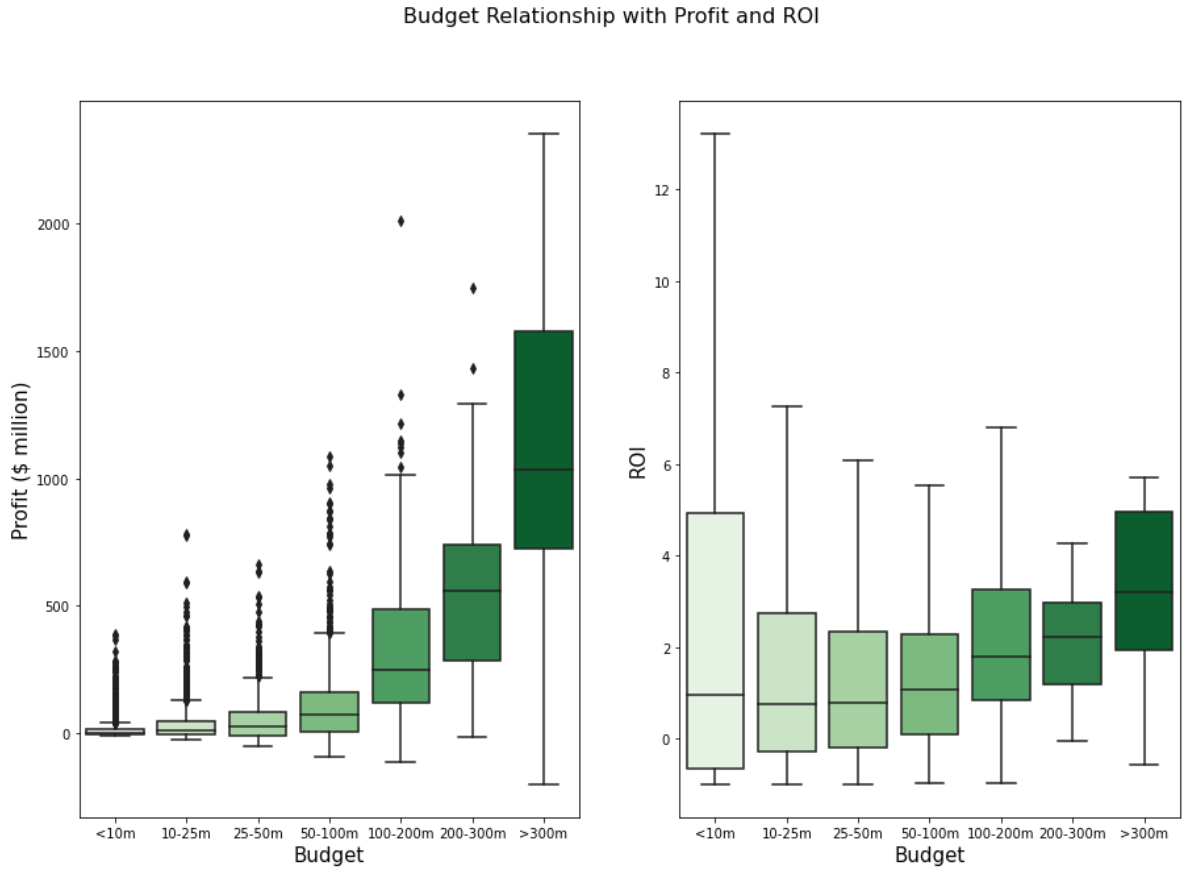
In [595]:

```
plt.figure(figsize = (10,7))
figure_5 = sns.barplot(y = 'p&l', x = 'budget_bin', data=budgets, color = 'mediumaquamarine
figure_5.set_xlabel('Budget ($)', fontsize=15)
figure_5.tick_params(labelsize=15, size=0)
figure_5.set_ylabel('%', fontsize=20)
figure_5.set_title('Percentage of Profitable Movies', fontsize=20);
```



In [596]:

```
figure_7, axes = plt.subplots(1,2, figsize=(15,10))
figure_7.suptitle('Budget Relationship with Profit and ROI', fontsize=16)
axes[0] = sns.boxplot(x = 'budget_bin', y = 'profit', palette="Greens", data = budgets, ax=
axes[1] = sns.boxplot(x = 'budget_bin', y = 'roi',palette="Greens", data = budgets, ax=axes
axes[1].set_ylabel('ROI', fontsize=15)
axes[0].set_ylabel('Profit ($ million)', fontsize=15)
axes[0].set_xlabel('Budget', fontsize = 15)
axes[1].set_xlabel('Budget', fontsize = 15);
```



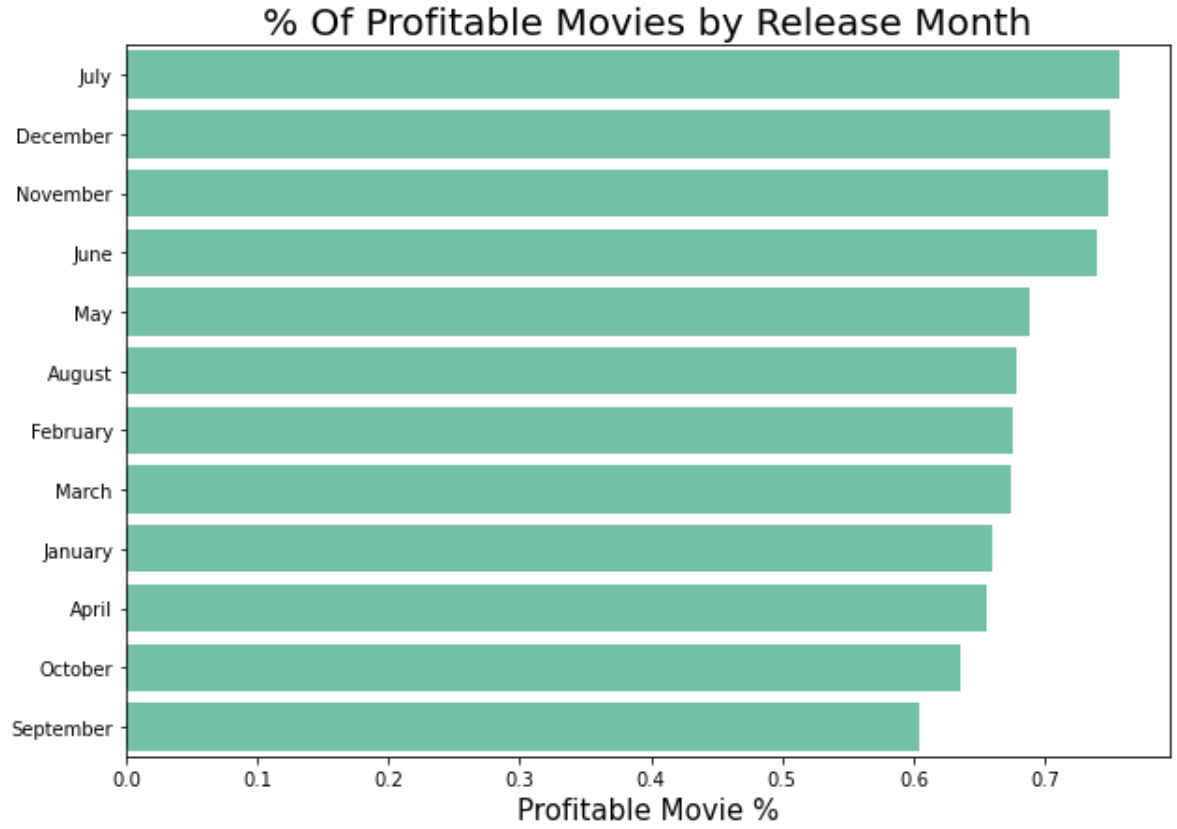
The most profitable budget is over 300m however this also seems to open up the possibility of incurring a loss as is shown by the lower whisker. Between 100m and 300m still has potential for large returns and typically out

performs lower budget ranges in terms of ROI, whilst spending more than this is still profitable more often than not as a company that is new to the business, I could not recommend spending more than $300m on the first movie.

When should you release a movie?

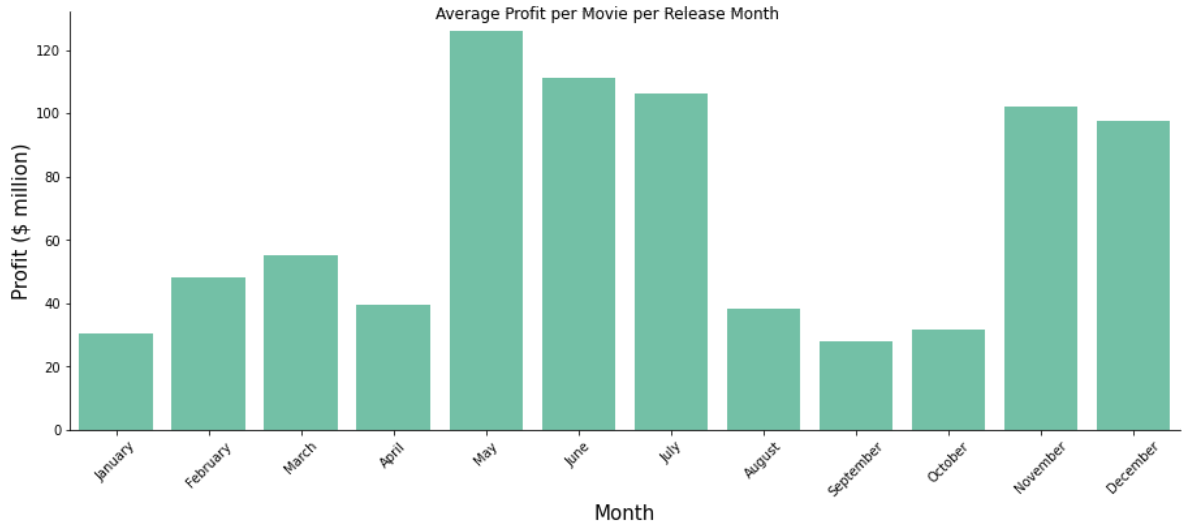
In [597]:

```
result = budgets.groupby('month').mean().reset_index().sort_values('p&l', ascending=False)
result.month = result['month'].replace([1,2,3,4,5,6,7,8,9,10,11,12],\
                                       ['January', 'February', 'March', 'April',\
                                       'May', 'June', 'July', 'August', 'September',\
                                       'October', 'November', 'December'])
plt.figure(figsize = (10,7))
figure_6 = sns.barplot( y = 'month', x = 'p&l', data=result, color = 'mediumaquamarine');
figure_6.set_title('% Of Profitable Movies by Release Month', fontsize=20);
figure_6.set_ylabel('', fontsize=15)
figure_6.set_xlabel('Profitable Movie %', fontsize=15);
```



In [598]:

```
figure_7 = sns.catplot(x = 'month', y='profit', kind= 'bar', color='mediumaquamarine',\
                       data = budgets[budgets['year']>1980], aspect = 2.5, ci=None)
figure_7.set_xticklabels(['January', 'February', 'March', 'April', 'May', 'June', 'July',\
                      'August', 'September', 'October', 'November', 'December'], rotation=4
figure_7.set_xlabels('Month', fontsize=15)
figure_7.set_ylabels('Profit ($ million)', fontsize=15)
figure_7.fig.suptitle('Average Profit per Movie per Release Month');
```



Summer and the run up to Christmas are the best times to release a movie in terms of % movies that are profitable and also in terms of the average profit per movie. Perhaps an exploration of Genre could provide a more definitive release month.

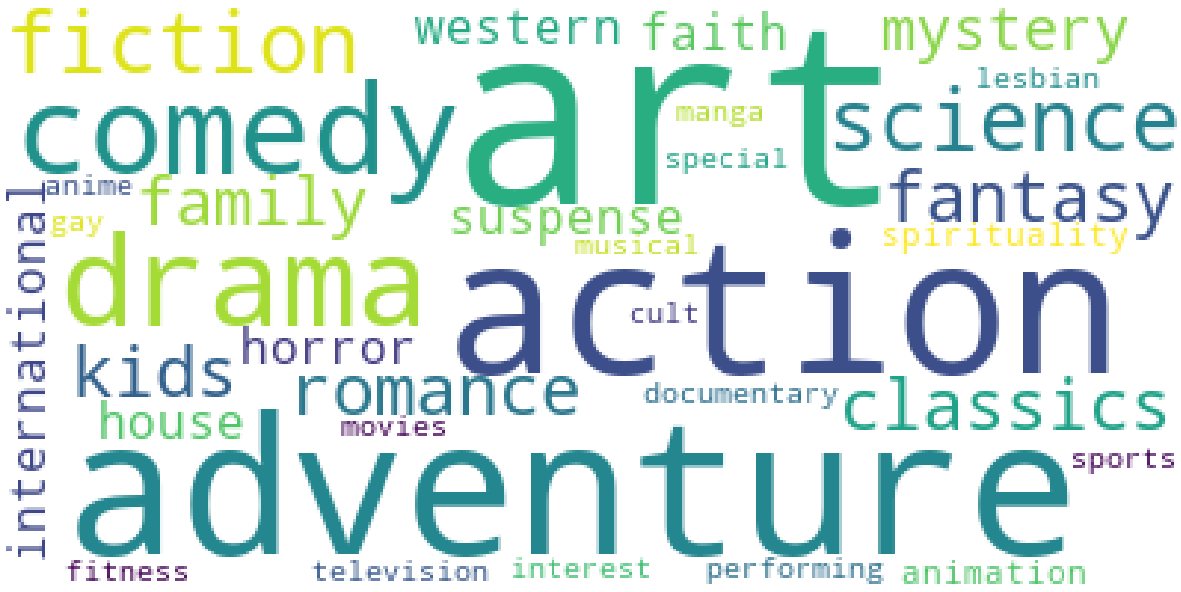
What Genre(s) should Microsoft Target?

The next question to answer is the type of movie that Microsoft should make, there are many genres to choose from...

In [599]:

```
from IPython.display import Image
Image(filename='./images/genres.png', width=900)
```

Out[599]:



In [600]:

```
df_info(genres_df)
```

```
################################################################################
########

genres_df Database


The length of this dataframe is 10095
Database shape (10095, 13)


<class 'pandas.core.frame.DataFrame'>
Int64Index: 10095 entries, 0 to 39648
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   primary_title      10095 non-null  object
 1   genre              10095 non-null  object
 2   release_date       10095 non-null  datetime64[ns]
 3   movie              10095 non-null  object
 4   production_budget  10095 non-null  float64
 5   domestic_gross     10095 non-null  float64
 6   worldwide_gross    10095 non-null  float64
 7   month              10095 non-null  float64
 8   year               10095 non-null  float64
 9   profit             10095 non-null  float64
 10  roi                10095 non-null  float64
 11  budget_bin         10095 non-null  category
 12  p&l                10095 non-null  float64
dtypes: category(1), datetime64[ns](1), float64(8), object(3)
memory usage: 1.0+ MB
None
```

	primary_title	genre	release_date	movie	production_budget	domestic_gross	wo
0	Percy Jackson & the Olympians: The Lightning T...	Action and Adventure	2010-02-12	Percy Jackson & the Olympians: The Lightning T...	95.0	88.768303	
1	Percy Jackson & the Olympians: The Lightning T...	Comedy	2010-02-12	Percy Jackson & the Olympians: The Lightning T...	95.0	88.768303	
2	Percy Jackson & the Olympians: The Lightning T...	Drama	2010-02-12	Percy Jackson & the Olympians: The Lightning T...	95.0	88.768303	

	primary_title	genre	release_date	movie	production_budget	domestic_gross	wo
3	Percy Jackson & the Olympians: The Lightning T...	Science Fiction and Fantasy	2010-02-12	Percy Jackson & the Olympians: The Lightning T...	95.0	88.768303	
4	Please Give	Comedy	2010-04-30	Please Give	3.0	4.033574	

```
Number of Null values in Genres_df

primary_title        0
genre                0
release_date         0
movie                0
production_budget    0
domestic_gross       0
worldwide_gross      0
month                0
year                 0
profit               0
roi                  0
budget_bin           0
p&l                  0
dtype: int64

The Number of duplicated rows in genres_df is 562
################################################################################
########
```

Profit and ROI are key for this analysis, therefore there is no value in imputing such a large number of missing values, insights wouldn't be accurate. Therefore I have already removed rows that had null values in what were the 'budgets' dataframe columns

To remind ourselves of the values available to us in this dataframe I will use .describe() once again

In [601]:

```
genres_df.describe()
```

Out[601]:

	production_budget	domestic_gross	worldwide_gross	month	year	
count	10095.000000	10095.000000	10095.000000	10095.000000	10095.00000	10095.
mean	41.126873	54.894091	122.241827	6.928876	2002.56424	81.
std	47.665314	77.796487	207.503723	3.422569	13.40801	175.
min	0.001100	0.000703	0.000703	1.000000	1915.00000	-200.
25%	9.200000	7.423977	11.773384	4.000000	1999.00000	-0.
50%	25.000000	29.580087	46.666955	7.000000	2005.00000	19.
75%	55.000000	67.631157	138.492394	10.000000	2011.00000	85.
max	425.000000	760.507625	2776.345279	12.000000	2019.00000	2351.

In [602]:

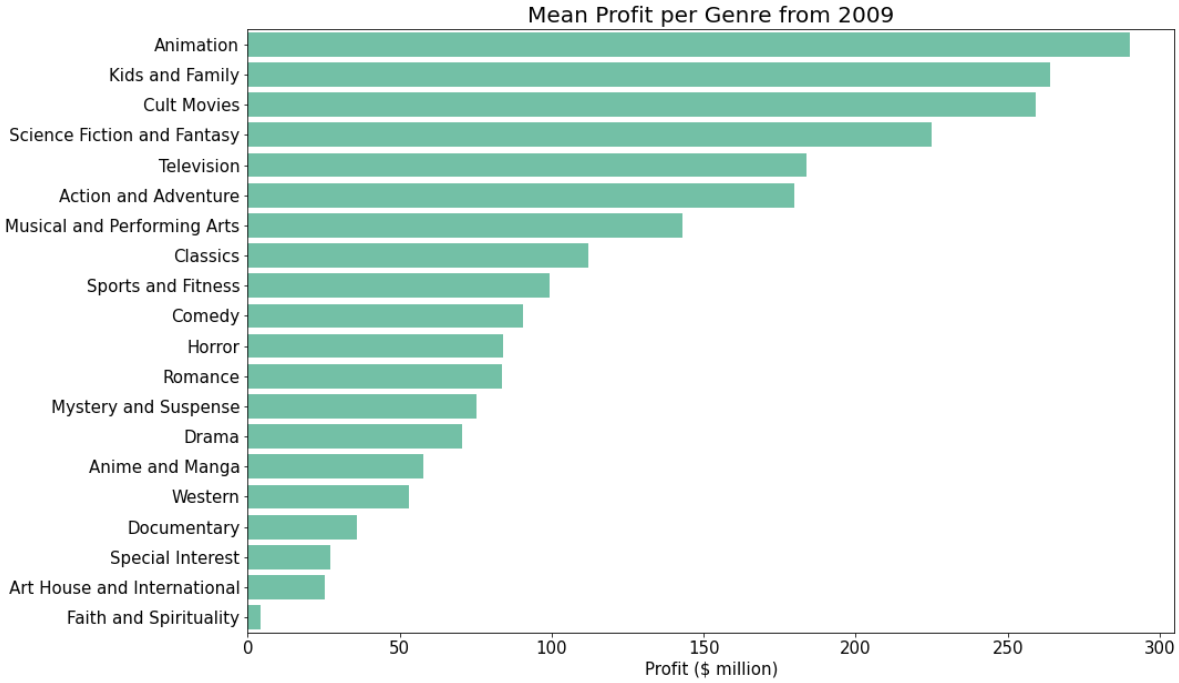
```
# new dataframe grouping median values by genre
new_genre_df = genres_df[genres_df['year']>2009].groupby(['genre']).mean().sort_values('pro
new_genre_df = new_genre_df.reset_index()
new_genre_df.head(25)
```

Out[602]:

	genre	production_budget	domestic_gross	worldwide_gross	month	year
0	Animation	104.217411	142.697327	394.560669	6.535714	2014.446429
1	Kids and Family	105.518440	134.895436	369.422066	6.645390	2013.907801
2	Cult Movies	41.833333	78.943991	300.932563	5.333333	2015.666667
3	Science Fiction and Fantasy	101.098704	115.786831	326.210171	6.148148	2014.062963
4	Television	55.800000	78.861047	239.590927	5.833333	2011.333333
5	Action and Adventure	87.906049	95.714938	267.699549	6.362140	2014.026749
6	Musical and Performing Arts	47.970408	85.486158	191.095841	7.591837	2013.469388
7	Classics	71.386842	66.337238	183.445306	7.447368	2013.736842
8	Sports and Fitness	38.750000	65.144939	138.017779	8.750000	2013.083333
9	Comedy	39.687832	56.574637	130.086144	6.372470	2013.331984
10	Horror	30.853533	45.792472	114.952606	6.396739	2014.005435
11	Romance	42.399745	51.532770	126.048246	6.080292	2012.788321
12	Mystery and Suspense	34.290293	43.847386	109.418108	6.599455	2013.444142
13	Drama	36.677989	45.257011	107.203068	7.008610	2013.713407
14	Anime and Manga	110.000000	40.563557	167.910690	3.000000	2017.000000
15	Western	54.161111	52.140796	107.050211	8.111111	2012.555556
16	Documentary	18.741528	25.962267	54.569469	6.805556	2012.666667
17	Special Interest	15.910833	18.391252	42.972229	7.500000	2012.200000
18	Art House and International	25.664310	17.147417	50.905220	6.327586	2012.137931
19	Faith and Spirituality	6.150000	10.162113	10.318816	8.000000	2012.166667

In [603]:

```
plt.figure(figsize = (15,10))
figure_10 = sns.barplot(y = 'genre', x = 'profit', data=new_genre_df, color = 'mediumaquama
figure_10.set_title('Mean Profit per Genre from 2009', fontsize=20);
figure_10.tick_params(labelsize=15)
figure_10.set_ylabel('')
figure_10.set_xlabel('Profit ($ million)', fontsize=15);
```



Ok, so animation is a clear winner in terms of average profit per movie, I suspect kids and family overlaps with animation, a lot of movies are likely to be in both genres.

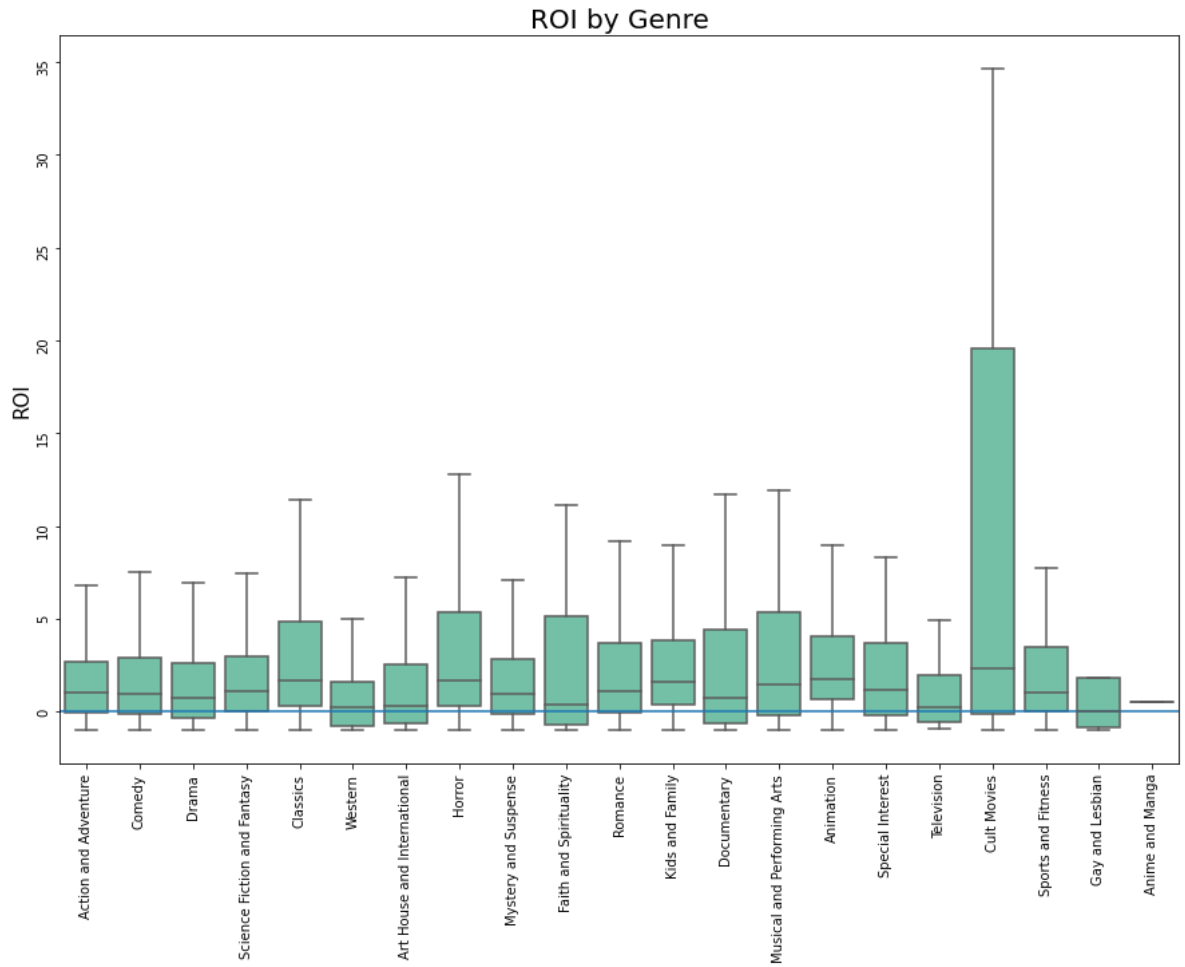
How do the genres look when we explore ROI?

In [604]:

```
plt.figure(figsize = (15,10))
figure_8 = sns.boxplot(y='roi', x = 'genre', color='mediumaquamarine', data= genres_df, sho
figure_8.set_title('ROI by Genre', fontsize = 20)
figure_8.tick_params(rotation=90)
figure_8.axhline()
figure_8.set_ylabel('ROI', fontsize=15)
figure_8.set_xlabel('')
```

Out[604]:

```
Text(0.5, 0, '')
```

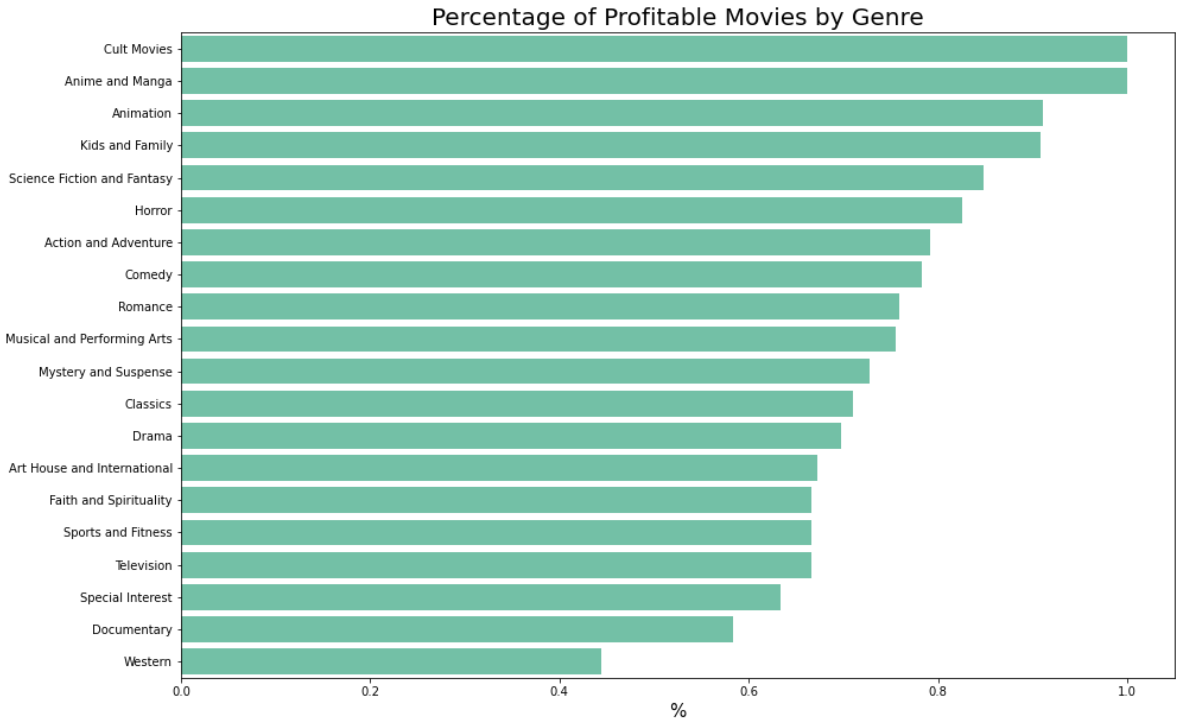


The recipe for making a cult classic is not written in stone and depends on many factors - as such it is not a genre I will explore.

This throws up some alternative suggestions to animation (which still performs strongly in ROI terms). Other genres that may be avenues to explore are Musicals and Horror. Faith and spirituality has a low median ROI value and its Lower quartile is below 0 making it more risky. One last thing to consider is the percentage of movies made in their respective genres that are profitable.

In [605]:

```
plt.figure(figsize = (15,10))
figure_9 = sns.barplot(y = 'genre', x = 'p&l', data=profit_loss, color = 'mediumaquamarine'
figure_9.set_title('Percentage of Profitable Movies by Genre', fontsize=20);
figure_9.set_ylabel('', fontsize=15)
figure_9.set_xlabel('%', fontsize=15);
```



Both Horror and Animation perform strongly in this metric too, therefore depending on the number of movies being made, the primary genre being recommended would be Animation. If Microsoft would like to spread their risk in multiple genres, Horror would be the second genre to prioritise.

Does Quality Matter?

If movie quality is an indicator of potential profits, this should be explored. Using Rotten Tomatoes dataframes it will be possible to plot movie rating against profit and ROI and determine whether there is a relationship, if it does matter what talent is best placed to deliver it?

WHAT IS THE AUDIENCE SCORE?

The Audience Score, denoted by a popcorn bucket, is the percentage of users who have rated the movie or TV Show positively. For films to which we can verify users have bought tickets, the Audience Score is made up of Verified Ratings.

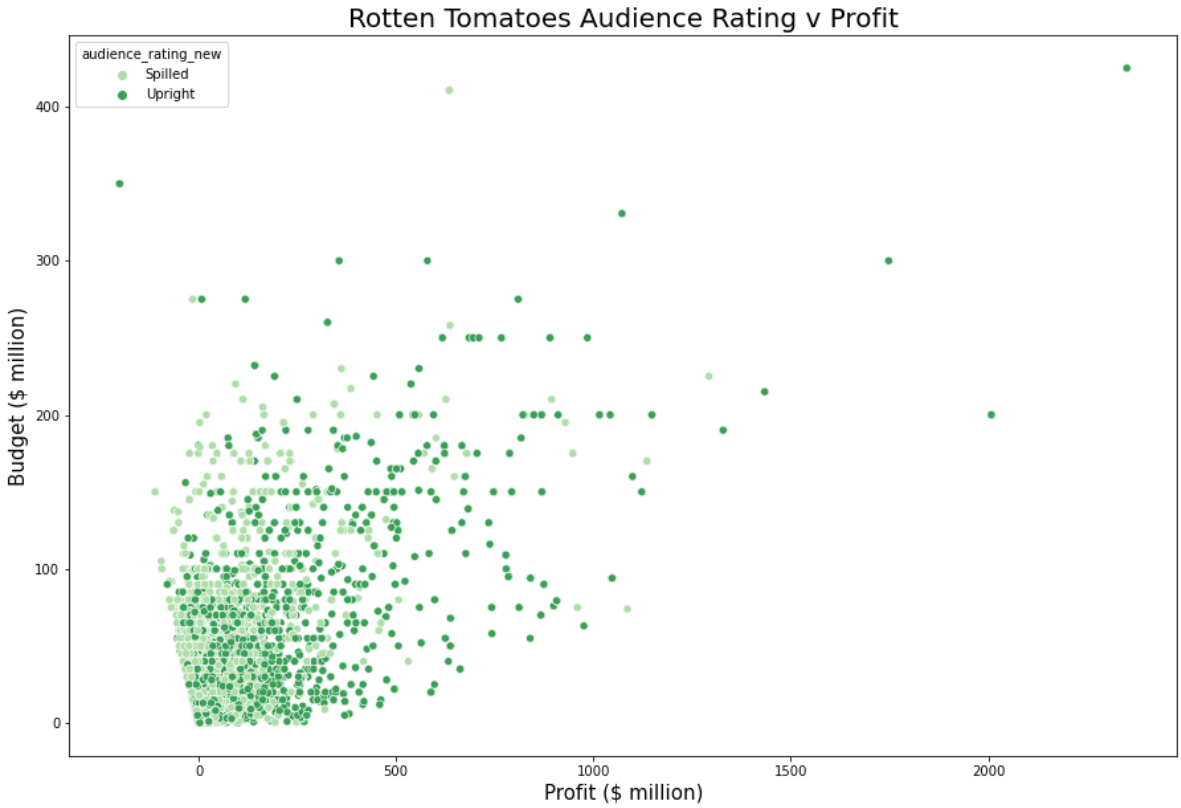
When at least 60% of users give a movie or TV show a star rating of 3.5 or higher, a full popcorn bucket is displayed to indicate its Fresh status.

When less than 60% of users give a movie or TV show a star rating of 3.5 or higher, a tipped over popcorn bucket is displayed to indicate its Rotten status.

Description of audience ratings taken from [Rotten Tomatoes (https://www.rottentomatoes.com/about)](https://www.rottentomatoes.com/about)

In [606]:

```
plt.figure(figsize = (15,10))
figure_11 = sns.scatterplot(data=ratings_profits, x="profit", y="production_budget",\
                            hue='audience_rating_new', palette='Greens')
figure_11.set_xlabel('Profit ($ million)', fontsize=15)
figure_11.set_ylabel('Budget ($ million)', fontsize=15);
figure_11.set_title('Rotten Tomatoes Audience Rating v Profit', fontsize = 20);
```



Audience rating seems to be a big factor when it comes to making profits, this makes sense, word of mouth is still valuable and people tend to listen to their friends and families when it comes to movie recommendations, I know I do.

When at least 60% of reviews for a movie or TV show are positive, a red tomato is displayed to indicate its Fresh status.

When less than 60% of reviews for a movie or TV show are positive, a green splat is displayed to indicate its Rotten status.

When there is no Tomatometer® score available, which could be because the Title hasn't released yet or there are not enough ratings to generate a score. New

WHAT IS CERTIFIED FRESH?

Certified Fresh status is a special distinction awarded to the best-reviewed movies and TV shows. In order to qualify, movies or TV shows must meet the following requirements:

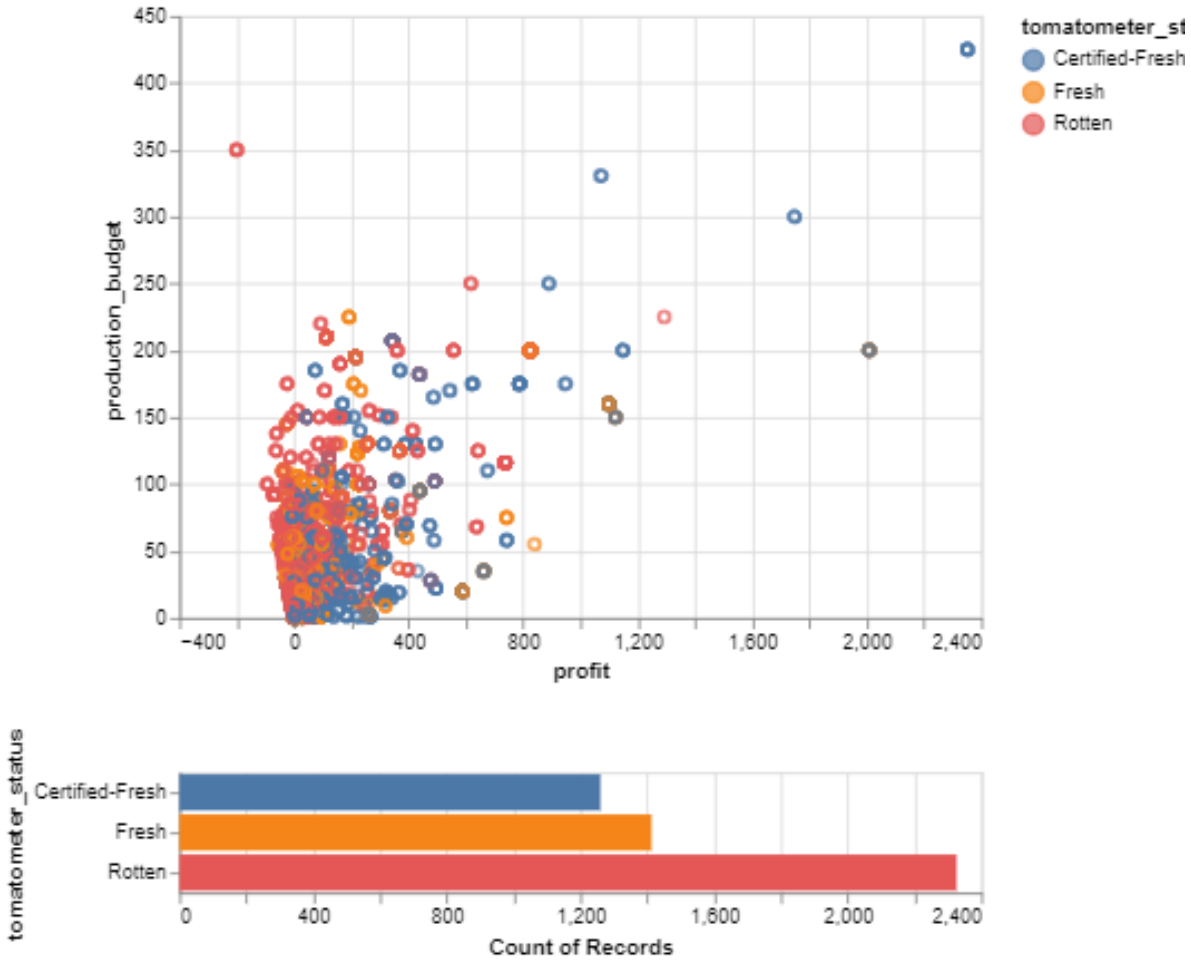
- A consistent Tomatometer score of 75% or higher.
- At least five reviews from Top Critics.
- Films in wide release must have a minimum of 80 reviews. This also applies for films going from limited to wide release.
- Films in limited release must have a minimum of 40 reviews.
- Only individual seasons of a TV show are eligible, and each must have a minimum of 20 reviews.

Description of Rotten Tomato ratings taken from [Rotten Tomatoes (https://www.rottentomatoes.com/about)](https://www.rottentomatoes.com/about)

In [607]:

```
import altair as alt


source = ratings_profits[:4999]

brush = alt.selection(type='interval')

points = alt.Chart(source).mark_point().encode(
    x='profit',
    y='production_budget',
    color=alt.condition(brush, 'tomatometer_status', alt.value('lightgray'))
).add_selection(
    brush
)

bars = alt.Chart(source).mark_bar().encode(
    y='tomatometer_status',
    color='tomatometer_status',
    x='count(tomatometer_status)'
).transform_filter(
    brush
)

points & bars
```

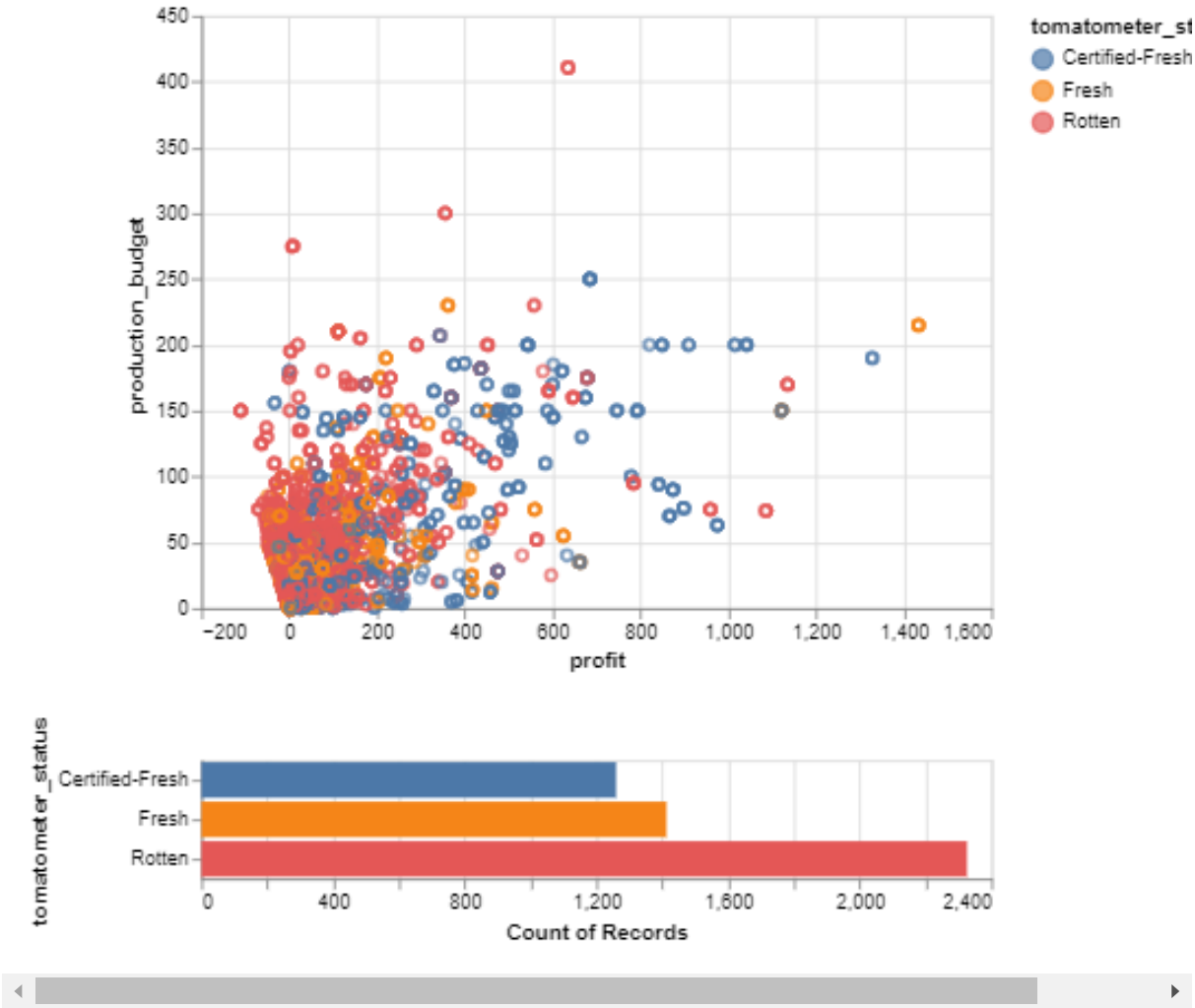
Out[607]:



In [608]:

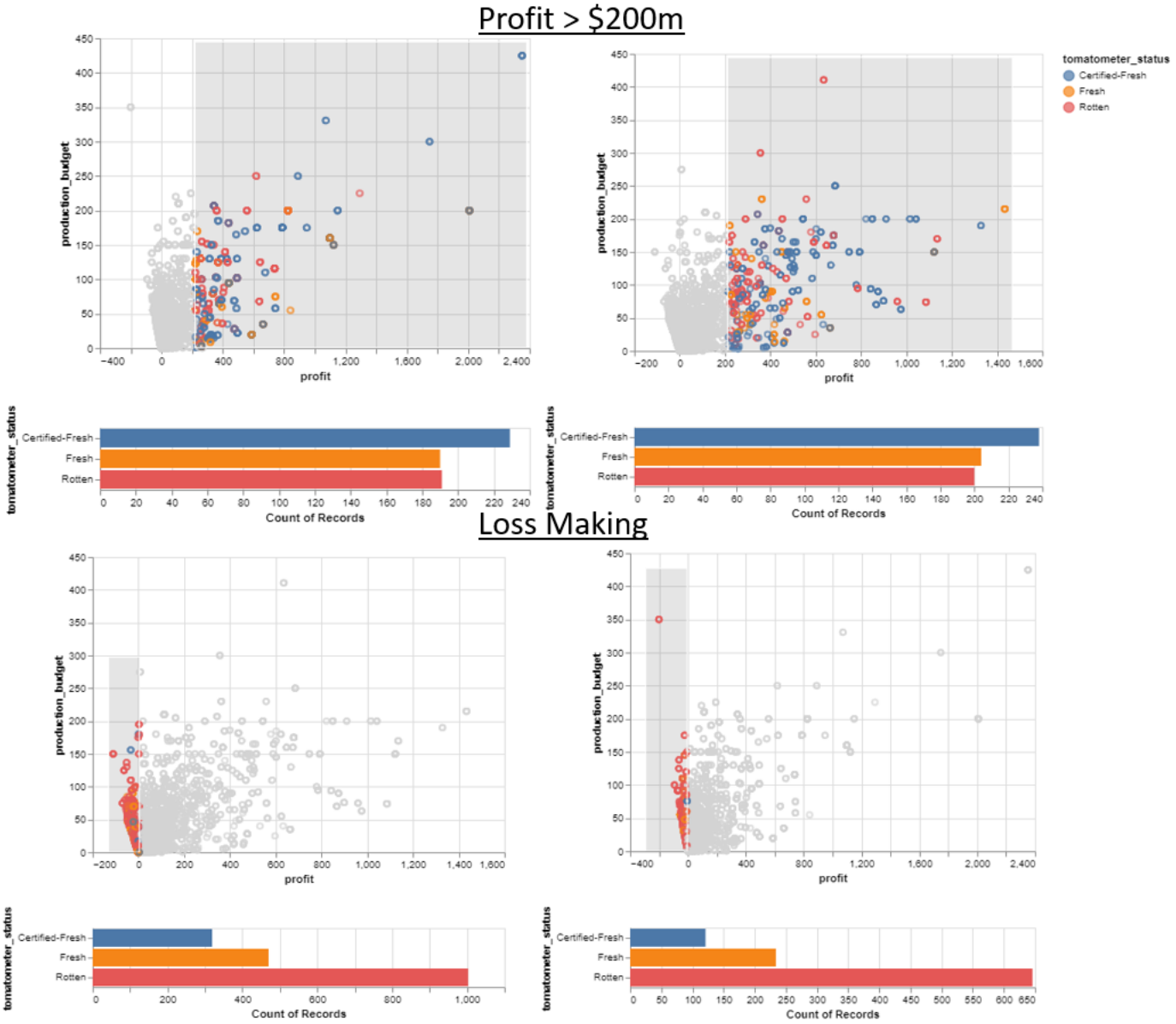
```
source2 = ratings_profits[5000:9999]

brush2 = alt.selection(type='interval')

points2 = alt.Chart(source2).mark_point().encode(
    x='profit',
    y='production_budget',
    color=alt.condition(brush, 'tomatometer_status', alt.value('lightgray'))
).add_selection(
    brush
)

bars2 = alt.Chart(source).mark_bar().encode(
    y='tomatometer_status',
    color='tomatometer_status',
    x='count(tomatometer_status)'
).transform_filter(
    brush
)

points2 & bars2
```

Out[608]:



Unfortunately, this visualisation technique is only usable on datasets that are less than 5000 rows, so I have broken the dataframe into 2, this provides data from 10,000 rows out of 12,524. This is enough to see a clear pattern, movies that are "Rotten" are much more likely to make a loss than those rated 'Certified Fresh' or

'Fresh'. So having a highly rated movie could help prevent losses, but isn't as strong an predictor of magnitude of profits.

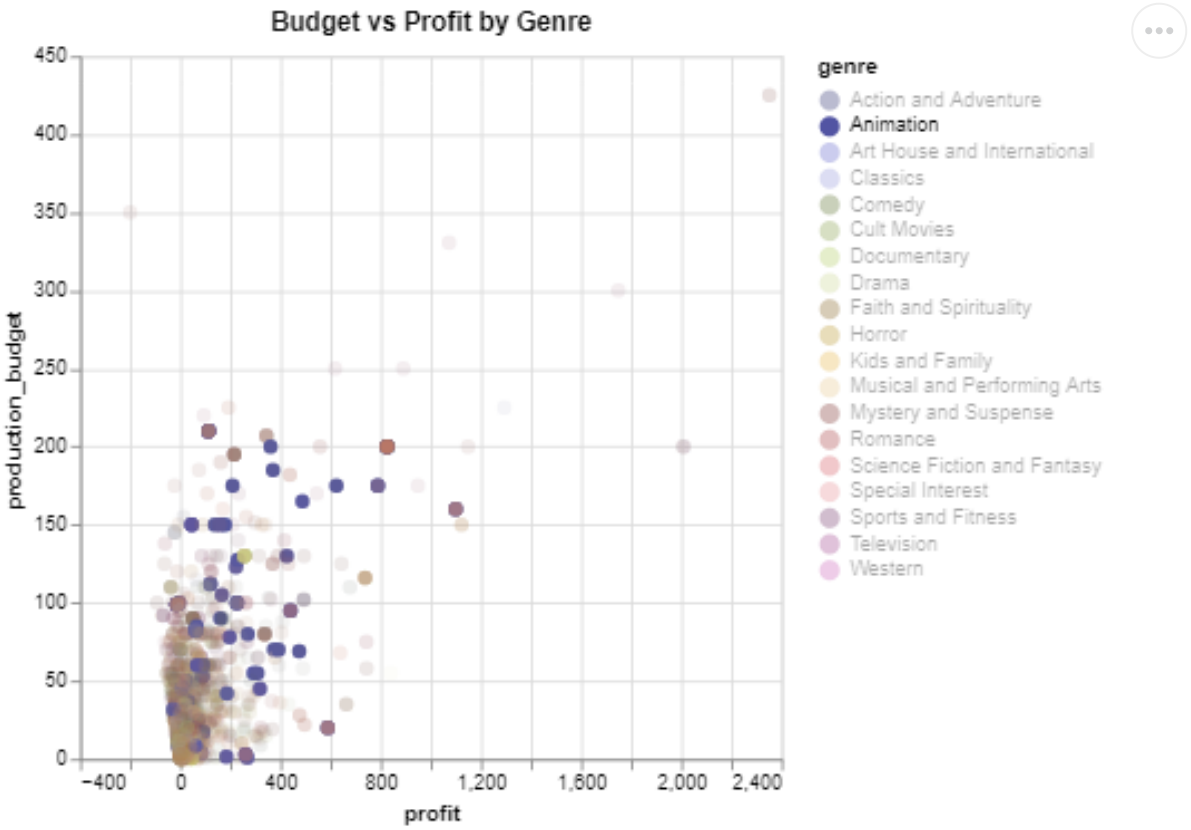


Unfortunately, this visualisation technique is only usable on datasets that are less than 5000 rows, so I have broken the dataframe into 2, this provides data from 10,000 rows out of 12,524. This is enough to see a clear pattern, movies that are "Rotten" are much more likely to make a loss than those rated 'Certified Fresh' or 'Fresh'. So having a highly rated movie could help prevent losses, but isn't as strong a predictor of magnitude of profits

In [609]:

```
# create scatter plot where the user can select and deselect genres using the legend
selection = alt.selection_multi(fields=['genre'], bind='legend')
alt.Chart(ratings_profits[0:5000]).mark_circle(size=50).\
encode(x='profit', y='production_budget', \
       color = alt.Color('genre:N', scale=alt.Scale(scheme='category20b')), tooltip=['movie
       opacity=alt.condition(selection, alt.value(1), alt.value(0.05))).properties\
(height=350, width=350, title='Budget vs Profit by Genre').add_selection(selection)
```

Out[609]:



Interesting insight into the budget for Animation movies can be drawn from this, previously believed the best option was between 100m - 300m, this plot indicates for the animation genre at least there are not many budgets that exceed $200m

Horror movies are less consistent with many making small losses.

Who makes Highly Rated Animation Movies?

It would be useful to have a shortlist of directors with experience of delivering highly rated animation movies. For this, I will revisit the rt_movies dataframe

In [610]:

```
rt_movies.head()
```

Out[610]:

	rotten_tomatoes_link	movie_title	content_rating	genres	directors	authors	
0	m/0814255	Percy Jackson & the Olympians: The Lightning T...	PG	Action & Adventure, Comedy, Drama, Science Fic...	Chris Columbus	Craig Titley, Chris Columbus, Rick Riordan	A
1	m/0878835	Please Give	R	Comedy	Nicole Holofcener	Nicole Holofcener	
2	m/10	10	R	Comedy, Romance	Blake Edwards	Blake Edwards	B
3	m/1000013-12_angry_men	12 Angry Men (Twelve Angry Men)	NR	Classics, Drama	Sidney Lumet	Reginald Rose	
4	m/1000079-20000_leagues_under_the_sea	20,000 Leagues Under The Sea	G	Action & Adventure, Drama, Kids & Family	Richard Fleischer	Earl Felton	

In [611]:

```
df_info(rt_movies)
```

```
################################################################################
########

rt_movies Database


The length of this dataframe is 17649
Database shape (17649, 16)


<class 'pandas.core.frame.DataFrame'>
Int64Index: 17649 entries, 0 to 17711
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   rotten_tomatoes_link   17649 non-null  object
 1   movie_title            17649 non-null  object
 2   content_rating         17649 non-null  object
 3   genres                 17649 non-null  object
 4   directors              17456 non-null  object
 5   authors                16118 non-null  object
 6   actors                 17298 non-null  object
 7   original_release_date  16507 non-null  object
 8   runtime                17365 non-null  float64
 9   production_company     17160 non-null  object
 10  tomatometer_status     17649 non-null  object
 11  tomatometer_rating     17649 non-null  float64
 12  tomatometer_count      17649 non-null  float64
 13  audience_rating        17649 non-null  float64
 14  audience_count         17649 non-null  float64
 15  audience_rating_new    17649 non-null  object
dtypes: float64(5), object(11)
memory usage: 2.3+ MB
None
```

	rotten_tomatoes_link	movie_title	content_rating	genres	directors	authors
0	m/0814255	Percy Jackson & the Olympians: The Lightning T...	PG	Action & Adventure, Comedy, Drama, Science Fic...	Chris Columbus	Craig Titley, Chris Columbus, Rick Riordan
1	m/0878835	Please Give	R	Comedy	Nicole Holofcener	Nicole Holofcener
2	m/10	10	R	Comedy, Romance	Blake Edwards	Blake Edwards

	rotten_tomatoes_link	movie_title	content_rating	genres	directors	authors
3	m/1000013-12_angry_men	12 Angry Men (Twelve Angry Men)	NR	Classics, Drama	Sidney Lumet	Reginald Rose
4	m/1000079-20000_leagues_under_the_sea	20,000 Leagues Under The Sea	G	Action & Adventure, Drama, Kids & Family	Richard Fleischer	Earl Felton

```
Number of Null values in Rt_movies

rotten_tomatoes_link        0
movie_title                 0
content_rating              0
genres                      0
directors                 193
authors                  1531
actors                    351
original_release_date    1142
runtime                   284
production_company        489
tomatometer_status          0
tomatometer_rating          0
tomatometer_count           0
audience_rating             0
audience_count              0
audience_rating_new         0
dtype: int64

The Number of duplicated rows in rt_movies is 0
################################################################################
########
```

In [612]:

```
top250 = animation_df.sort_values('audience_rating', ascending=False)[0:250]
top250.dropna(subset=['directors'], inplace=True)
top250.dropna(subset=['actors'], inplace=True)
```

In [613]:

```
top250.head()
```

Out[613]:

	rotten_tomatoes_link	movie_title	content_rating	genres	directors	authors	act
12532	m/ride_your_wave	Ride Your Wave	NR	Animation, Anime & Manga, Comedy, Romance	Masaaki Yuasa	Reiko Yoshida, Stephanie Sheh	Ry Katayc Rina Kaw Hon Matsumot
10014	m/man_in_camo	Man in Camo	NR	Animation, Documentary	Ethan H. Minsker	NaN	Ethar Minsker, Butler, La Flook, T
410	m/1006223-how_the_grinch_stole_christmas	How the Grinch Stole Christmas	NR	Animation, Classics, Comedy, Kids & Family, Mu...	Chuck Jones, Ben Washam	Dr. Seuss	Boris Kar June Fo T Ravensc
				Animation			Ja

In [614]:

```
top250.production_company.value_counts()
```

Out[614]:

```
Walt Disney Pictures           27
GKIDS                          16
20th Century Fox               11
Buena Vista Pictures           11
Warner Bros. Pictures          10
                               ..
Rita Productions                1
Hannover House                  1
Sony Pictures Entertainment     1
DisneyToon Studios              1
United Artists                  1
Name: production_company, Length: 101, dtype: int64
```

Walt Disney, unsurprisingly top the number of movies in the 250 highest rated Animation movies...there may be something that can be learned from them

In [615]:

```
top250_directors = dict(splitter_counter(top250, 'directors'))
for key, value in top250_directors.items():
    if value > 1:
        print(key, value)
```

```
C:\Users\Andrew\anaconda3\envs\learn-env\lib\site-packages\pandas\core\strin
gs.py:2001: UserWarning: This pattern has match groups. To actually get the
groups, use str.extract.
  return func(self, *args, **kwargs)

Hayao Miyazaki 10
John Musker 6
Ron Clements 6
Wolfgang Reitherman 6
Clyde Geronimi 6
Jay Oliva 5
Hamilton Luske 5
Wilfred Jackson 5
Lee Unkrich 4
Mamoru Hosoda 4
John Lasseter 4
Dean DeBlois 4
Andrew Stanton 4
Brad Bird 4
Bill Melendez 4
Isao Takahata 3
Chris Buck 3
Tomm Moore 3
Byron Howard 3
Satoshi Kon 3
Chris Sanders 3
Don Hall 3
Pete Docter 3
Chris Renaud 3
Bill Plympton 3
Sam Liu 3
Masaaki Yuasa 2
Chuck Jones 2
Dan Scanlon 2
Peter Ramsey 2
Roger Allers 2
Rob Minkoff 2
Jennifer Lee 2
Rich Moore 2
Henry Selick 2
Tim Burton 2
Jan Svankmajer 2
David Silverman 2
Sylvain Chomet 2
Burny Mattinson 2
Mamoru Oshii 2
Martin Rosen 2
Hiromasa Yonebayashi 2
Vincent Patar 2
Stéphane Aubier 2
Richard Linklater 2
Nora Twomey 2
Wes Anderson 2
Chris McKay 2
```

```
Ethan Spaulding 2
Arthur Rankin Jr. 2
Jules Bass 2
Jake Castorena 2
Don Bluth 2
Nina Paley 2
Barry Cook 2
Pierre Coffin 2
Angus MacLane 2
Richard Starzak 2
Stephen J. Anderson 2
Ralph Bakshi 2
Lauren Montgomery 2
Jennifer Yuh Nelson 2
Hiroyuki Okiura 2
Brenda Chapman 2
Steve Martino 2
Bradley Raymond 2
Gaëtan Brizzi 2
Paul Brizzi 2
Joann Sfar 2
```

Looking at this list, it requires some research into Directors whom are still alive. Focusing on the top 250, I would recommend approaching John Musker, Brad Bird, Dean DeBlois, Jennifer Lee.

They are responsible for hits such as Moana, The Incredibles, Lilo & Stitch and Frozen.

It may not be possible to pull this talent away from Disney, further work would be required on searching for a Director that is available and has a proven track record.

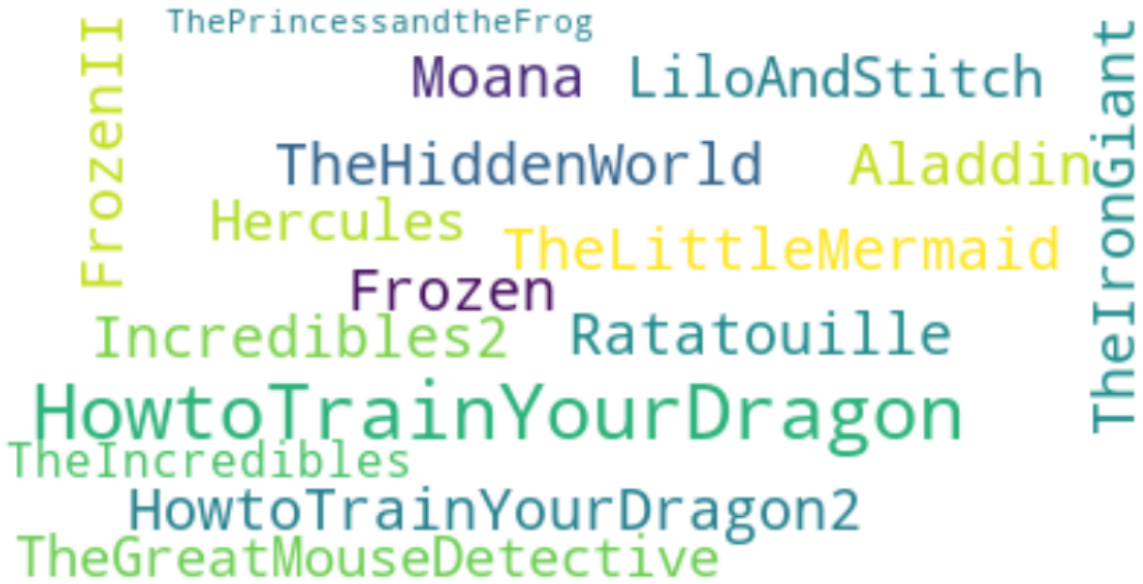
In [616]:

```
# Create list of movies that directors I have shortlisted have made

directors_shortlist = ['Brad Bird', 'Dean DeBlois', 'Jennifer Lee', 'John Musker']
table_directors = top250[(top250['directors'].str.contains('Dean DeBlois'))\
       | (top250['directors'].str.contains('Jennifer Lee'))\
       | (top250['directors'].str.contains('Brad Bird'))\
       | (top250['directors'].str.contains('John Musker'))]
table_directors[['movie_title','directors','audience_rating']]

unwanted_directors = []
new_directors = []
for entry in table_directors['directors']:
    #print(entry)
    for i in entry.split(','):
        i = i.strip()
        if i in directors_shortlist:
            new_directors.append(i)
        elif i not in directors_shortlist:
            unwanted_directors.append(i)
table_directors['directors'] = new_directors
```

```
<ipython-input-616-e2c8a1df79fa>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  table_directors['directors'] = new_directors
```

In [617]:

```
### create wordcloud of hit movies they have made

plt.figure(figsize=[15,15])
# join the list and lowercase all the words
list_movies = []

for entry in table_directors['movie_title']:
    entry = entry.replace('&', 'And')
    list_movies.append(entry.replace(' ', ''))

text2 = ' '.join(list_movies)

#create the wordcloud object
wordcloud2 = WordCloud(collocations=True, background_color='white').generate(text2)

#wordcloud2.to_file("./images/hitmovies.png");
#plot the wordcloud2 object
plt.imshow(wordcloud2, interpolation='bilInear')
plt.axis('off')
plt.show();
```



In [618]:

```
top250_actors = dict(splitter_counter(top250, 'actors'))

## check to see multiple entries in top 250 highest rated animation films

for key, value in top250_actors.items():
    if value > 3:
        print(key, value)
```

```
John Ratzenberger 17
Tom Hanks 8
John DiMaggio 8
Joan Cusack 7
Tim Allen 7
Bonnie Hunt 7
Alan Tudyk 7
Edie McClurg 7
Lucy Liu 7
Wallace Shawn 6
Jack Angel 6
Cary Elwes 6
Jonah Hill 6
Fred Tatasciore 6
Tara Strong 5
Crispin Freeman 5
Kristen Schaal 5
Don Rickles 5
Koichi Yamadera 5
Jennifer Darling 5
Joe Ranft 5
Craig Ferguson 5
Kristen Wiig 5
Jess Harnell 5
Jan Rabson 5
Teddy Newton 5
Laraine Newman 5
Bob Peterson 5
Pamela Adlon 5
Barbara Luddy 5
Brad Garrett 5
Mona Marshall 5
Will Arnett 5
Mae Whitman 5
Candy Candido 5
June Foray 4
Rashida Jones 4
Ryûnosuke Kamiki 4
Cheech Marin 4
Jodi Benson 4
Minnie Driver 4
Angela Lansbury 4
David Ogden Stiers 4
Kristen Bell 4
Alfred Molina 4
Frank Welker 4
Charlie Adler 4
Sherry Lynn 4
Mickie T. McGowan 4
Patrick Pinney 4
Nathan Fillion 4
```

```
Laurie Metcalf 4
Mark Hamill 4
Tress MacNeille 4
America Ferrera 4
Jeff Pidgeon 4
John Goodman 4
Frank Oz 4
Bob Bergen 4
Richard Kind 4
Bill Hader 4
Scott Menville 4
Kevin Conroy 4
Rene Auberjonois 4
Dave Foley 4
Grey DeLisle 4
Terri Douglas 4
Bryan Cranston 4
Wayne Knight 4
Andrew Stanton 4
Bill Melendez 4
Kevin Michael Richardson 4
Stephen Root 4
Rosario Dawson 4
James Hong 4
Jack Black 4
Seth Rogen 4
Tom Kenny 4
Raven-Symoné 4
Kristin Chenoweth 4
```

As is to be expected, for animated movies, the actors associated with high audience rating movies are not household names with perhaps the exception of Tom Hanks, it will be interesting if this is the same case for the most profitable animated movies...

In [619]:

```
# creating dataframe for most profitable animated movies

top_25 = animation_by_profit[0:25]
top_25_actors = dict(splitter_counter(top_25, 'actors'))

# check to see multiple entries for actors in top 25 most profitable animated films
for key, value in top_25_actors.items():
    if value > 2:
        print(key, value)
```

```
Steve Coogan 4
James Earl Jones 4
Whoopi Goldberg 4
Josh Gad 3
Alan Tudyk 3
Jesse Corti 3
Chris Renaud 3
Michael Beattie 3
Allison Janney 3
Steve Carell 3
Pierre Coffin 3
Jenny Slate 3
Idris Elba 3
John Ratzenberger 3
Albert Brooks 3
Sterling Holloway 3
```

Conclusions & Further Work

- Movies are making more profits now than at any other point in history, however, budgets have increased during that time which reduces the Return on Investment.
- The Animation genre has the highest percentage of profitable movies of any other genres and this would be the first area in which I would recommend Microsoft's focused. Animation genre could be broken down into many sub-genres and this would be a line of work that needs further study.
- Horror is a genre which can reap big rewards, particularly in terms of ROI. It is not as consistently profitable as movies in the animation genre but is certainly worth exploring further.
- The recommended month for release would be July, the data shows July has the highest percentage of profitable movies than all others. Not only that, movies released in May / June / July and November / December are the most profitable. The reason for this should be investigated, it would be interesting to see if there is a correlation between school holidays and movie success and if this is impacted by the rating (PG-13, PG etc.) that the movie is designated. This recommendation is preliminary, release month by genre should be explored.
- It would be recommended to invest up to 200m U.S dollars on an animation movie, a Return of Investment of 2 is a reasonable target, i.e $400m profit.
- Rotten Tomatoes rating vs Profits show the majority of loss making movies have received poor reviews, this indicates achieving positive reviews could be important in determining the success of a movie.

- A better understanding of what is covered under the "budget" term, does this include marketing? If not this is something else which will eat into profits that is not accounted for here.
- There is a moderate positive correlation between budget and profit, this does not prove causality and spending money is no guarantee on the ROI (no correlation between budget and ROI), money needs to be spent in the correct manner on getting the right people - preferably talent that has a proven track record of delivering high quality movies.
- In terms of directors, John Musker, Brad Bird, Dean DeBlois, Jennifer Lee would near the top of any list to be approached. All have been in charge of delivering successful animated movies.
- Since this is an animated movie, I have not fully explored options for lead actor roles. I would imagine the influence a lead actor has on movie profits would be dampened in the Animation genre when compared to other live action genres - this would need to be confirmed. There are still many famous names that appear in successful animated films, actors with a background in comedy is particularly common.
- I would recommend a standalone study of the potential future impact streaming companies and platforms could have on future movies' profitabilty and an analysis of the number of people visiting movie theatres.
- Further work is recommended on analysing big budget failures of the last 10 years, how and why did they fail?
- Investigate additional revenue streams - merchandise and games, both could be lucrative and Microsoft would be well placed to take advantage in the gaming sector.

In [ ]: