Final Project Submission

Please fill out:

- Student name: Stephen Gomes
- Student pace: self paced
- Scheduled project review date/time: 2022.10.26 15:00
- Instructor name: Joe Comeaux
- Blog post URL: https://github.com/steve-gomes/steve-gomes.github.io/blob/main/_posts/2022-10-15-your-new-blog-post.md

Current Movie Industry Analysis

Author: Stephen Gomes

Summary

The task assigned is to advise Microsoft on their launch of a new movie studio. They are in need of insight as to the type of movies to produce in order to be most successful at the box office. There are many levers we can pull when deciding on making a movie - total budget, genre of movie, staff (directors, writers, actors, run length, release date, etc.

After exploration I focussed on budget, genre, run length and release scheduling as the main drivers.

The recommendations I make based on data analysis are:

- Focus your studio on Action & Adventure movies as these genres are the most profitable
- Release films primarily in Winter & Summer seasons, as box office sales show moviegoing is highest in these seasons
- Budget up to 300M per film, as ROI and Profit increases with budget until 300M at which point it drops off
- Keep movie runtime in the sweet spot of 90-120 minutes length as higher run times do not result in higher sales

Business Problem

Microsoft is launching a new business studio and in need of insight as to the types of movies to launch to maximize their successs at the box office. In answering the question of what

maximizes success, I focused on revenue & ROI as the measures of success, and did not use any movie ratings. The thesis I had was that while having well rated movies is nice, it is not something you can directly control for like genre, staffing, budget, or release date. Similarly, the ultimate measure of success in business is sales, not product reviews.

Data Inputs

For this analysis I used the following datasets & sources:

IMDB - Reference data for movie genre

Rotten Tomatoes - Reference data for movie release date, length, writers, diretors, plus box office sales

The Numbers - Movie budget data

Minneapolis Fed - Inflation data

Basics - imports & file loading

In [87]:
```
# Your code here - remember to use markdown cells for comments as well!

# setup all our imports
import json
import pandas as pd
import numpy as np
import matplotlib
import sqlite3
import requests
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter

%matplotlib inline

# clear some warnings on edits to copies of dataslice
pd.options.mode.chained_assignment = None  # default='warn'
```

In [88]:
```
# read in all our input files
info = pd.read_csv('zippedData/rt.movie_info.tsv.gz', sep="\t") # USED
budgets = pd.read_csv('zippedData/tn.movie_budgets.csv.gz') # USED
# open DB file for query, get list of tables, USED
conn = sqlite3.connect('zippedData/im.db')
tbls = pd.read_sql("""SELECT name FROM sqlite_master WHERE type = 'table';"""

# SUPPLIED INPUTS NOT USED BELOW
# reviews file had a utf8 encoding error with default pandas read encoding
reviews = pd.read_csv('zippedData/rt.reviews.tsv.gz', sep="\t", encoding = "I
bom = pd.read_csv('zippedData/bom.movie_gross.csv.gz') # not used
tmdb = pd.read_csv('zippedData/tmdb.movies.csv.gz') # not used
```

In [89]:
```
tbls
```

Out[89]:

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

In [90]:
```
# select info from movie_basics table to enrich budgets with genre
movie_basics = pd.read_sql("""SELECT primary_title AS movie,start_year AS yea
conn.close()
```

Introducing a new dataset - CPI

We are comparing & aggregating dollar figures across time for budgets and box office sales. When comparing dollar figures it is important to use "real" not "nominal" numbers, and therefore to inflation adjust across years to a fixed year. In this case we chose current year, 2022, as the baseline. So we use inflation data from the Minneapolis Fed, who have a convenient consumer price index time series going back to the early 1900s.

In [91]:
```
# we need to adjust historical dollars to 2022 level, let's get CPI data from
# webscrape directly with pandas & requests
# grab CPI of Minn. Fed site
html_page = requests.get('https://www.minneapolisfed.org/about-us/monetary-po
webdf_list = pd.read_html(html_page.text) # pull the table from the html
webdf = webdf_list[0]
webdf
```

Out[91]:

	Year	Annual Average CPI(-U)	Annual Percent Change (rate of inflation)
0	1913	9.9	NaN
1	1914	10.0	1.3%
2	1915	10.1	0.9%
3	1916	10.9	7.7%
4	1917	12.8	17.8%
...	...	...	...
105	2018	251.1	2.4%
106	2019	255.7	1.8%
107	2020	258.8	1.2%
108	2021	271.0	4.7%
109	2022*	294.4	8.6%

110 rows × 3 columns

Data cleaning & manipulation

CPI data

Using the CPI timeseries, we create a multiplier that adjusts each year to the 2022 dollar figure. We use this later on all tables containing dollar figures.

In [92]:
```
# cleanup CPI data
# rename CPI column, remove inflation rate, create annual multiplier column w
cpi = webdf.rename({'Year' : 'year' , 'Annual Average CPI(-U)': 'CPI'}, axis=
cpi = cpi.iloc[: , :-1]
cpi['CPI_mult'] = cpi.iloc[-1]['CPI':].div(cpi.loc[:,'CPI':]).astype(np.float
# clean up current year * label, cast to int
cpi.loc[cpi['year'] == '2022*', 'year'] = '2022'
cpi['year'] = cpi['year'].astype(np.int64)
cpi
```

Out[92]:

	year	CPI	CPI_mult
0	1913	9.9	29.737374
1	1914	10.0	29.440000
2	1915	10.1	29.148515
3	1916	10.9	27.009174
4	1917	12.8	23.000000
...	...	...	...
105	2018	251.1	1.172441
106	2019	255.7	1.151349
107	2020	258.8	1.137558
108	2021	271.0	1.086347
109	2022	294.4	1.000000

110 rows × 3 columns

Budget table

Using the provided movie budget data from The Numbers we do the following

- Cleanup data types for date and dollar figures
- Join on movie_basics table from IMDB, which contains title->genre mapping
- Remove rows we cannot find a genre for
- Join on CPI data from MN Fed, and adjust the dollar figure columns in new 2022 tagged columns
- Compute ROI
- Bucket budgets into 100M buckets

In [93]:
```
budgets
```

Out[93]:

	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
...	...	...	...	...	...	...
5777	78	Dec 31, 2018	Red 11	$7,000	$0	$0
5778	79	Apr 2, 1999	Following	$6,000	$48,482	$240,495
5779	80	Jul 13, 2005	Return to the Land of Wonders	$5,000	$1,338	$1,338
5780	81	Sep 29, 2015	A Plague So Pleasant	$1,400	$0	$0
5781	82	Aug 5, 2005	My Date With Drew	$1,100	$181,041	$181,041

5782 rows × 6 columns

In [94]:
```
movie_basics
```

Out[94]:

	movie	year	genres
0	Sunghursh	2013	Action,Crime,Drama
1	One Day Before the Rainy Season	2019	Biography,Drama
2	The Other Side of the Wind	2018	Drama
3	Sabse Bada Sukh	2018	Comedy,Drama
4	The Wandering Soap Opera	2017	Comedy,Drama,Fantasy
...	...	...	...
146139	Kuambil Lagi Hatiku	2019	Drama
146140	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	Documentary
146141	Dankyavar Danka	2013	Comedy
146142	6 Gunn	2017	None
146143	Chico Albuquerque - Revelações	2013	Documentary

146144 rows × 3 columns

In [95]:

```
# cleanup budgets data
clean_budgets = budgets
clean_budgets['release_date'] = pd.to_datetime(clean_budgets['release_date'])
clean_budgets['year'] = clean_budgets['release_date'].dt.year
clean_budgets = clean_budgets.merge(movie_basics,on=['year','movie'],how='lef
clean_budgets = clean_budgets.loc[~clean_budgets['genres'].isnull()]
# join CPI info & inflate $ to 2022 figures
clean_budgets = clean_budgets.merge(cpi, on='year', how='left')
clean_budgets["worldwide_gross"] = clean_budgets["worldwide_gross"].replace("
clean_budgets["production_budget"] = clean_budgets["production_budget"].repla
clean_budgets['gross2022'] = clean_budgets['worldwide_gross'] * clean_budgets
clean_budgets['budget2022'] = clean_budgets['production_budget'] * clean_budg
clean_budgets['net2022'] = clean_budgets['gross2022'] - clean_budgets['budget
clean_budgets["roi"] = 100 * clean_budgets["net2022"] / clean_budgets["budget

# bin movie budgets into 50M buckets
bins = [0, 0.5e8, 1e8, 1.5e8, 2e8, 2.5e8, 3e8, 3.5e8, 4e8, 4.5e8, 5e8, 5.5e8,
blabel = ['50M', '100M', '150M', '200M', '250M', '300M', '350M', '400M', '450


clean_budgets['budget2022bin'] = pd.cut(clean_budgets['budget2022'], bins,lab

clean_budgets
```

Out[95]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	yea
0	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000.0	$241,063,875	1.045664e+09	201
1	3	2019-06-07	Dark Phoenix	350000000.0	$42,762,350	1.497624e+08	201
2	4	2015-05-01	Avengers: Age of Ultron	330600000.0	$459,005,868	1.403014e+09	201
3	7	2018-04-27	Avengers: Infinity War	300000000.0	$678,815,482	2.048134e+09	201
4	9	2017-11-17	Justice League	300000000.0	$229,024,295	6.559452e+08	201
...	...	...	...	...	...	...	.
1536	45	2017-01-27	Emily	27000.0	$3,547	3.547000e+03	201
1537	49	2015-09-01	Exeter	25000.0	$0	4.897920e+05	201
1538	52	2015-12-01	Dutch Kills	25000.0	$0	0.000000e+00	201
1539	59	2011-11-25	The Ridges	17300.0	$0	0.000000e+00	201
1540	62	2014-12-31	Stories of Our Lives	15000.0	$0	0.000000e+00	201

1541 rows × 15 columns

Clean Budgets check & remove dupes

In [111…

```
clean_budgets.describe()
```

Out[111…

	id	production_budget	worldwide_gross	year	CPI	CPI_m
count	1541.000000	1.541000e+03	1.541000e+03	1541.000000	1541.000000	1541.000(
mean	50.609994	4.439437e+07	1.392236e+08	2013.889682	234.692992	1.2564
std	28.861814	5.587520e+07	2.328518e+08	2.574758	9.583139	0.051:
min	1.000000	1.500000e+04	0.000000e+00	2010.000000	218.100000	1.137:
25%	26.000000	8.000000e+06	7.313697e+06	2012.000000	229.600000	1.226(
50%	51.000000	2.250000e+07	5.033442e+07	2014.000000	236.700000	1.243:
75%	76.000000	5.500000e+07	1.565536e+08	2016.000000	240.000000	1.282:
max	100.000000	4.106000e+08	2.048134e+09	2020.000000	258.800000	1.349(

In [112…

```
clean_budgets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1541 entries, 0 to 1540
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype         
---  ------             --------------  -----         
 0   id                 1541 non-null   int64         
 1   release_date       1541 non-null   datetime64[ns]
 2   movie              1541 non-null   object        
 3   production_budget  1541 non-null   float64       
 4   domestic_gross     1541 non-null   object        
 5   worldwide_gross    1541 non-null   float64       
 6   year               1541 non-null   int64         
 7   genres             1541 non-null   object        
 8   CPI                1541 non-null   float64       
 9   CPI_mult           1541 non-null   float64       
 10  gross2022          1541 non-null   float64       
 11  budget2022         1541 non-null   float64       
 12  net2022            1541 non-null   float64       
 13  roi                1541 non-null   float64       
 14  budget2022bin      1541 non-null   category      
dtypes: category(1), datetime64[ns](1), float64(8), int64(2), object(3)
memory usage: 222.5+ KB
```

In [116…

```
# remove dupes
clean_budgets = clean_budgets.drop_duplicates()
clean_budgets.info()
# removes 4 rows
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1537 entries, 0 to 1540
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype         
---  ------             --------------  -----         
 0   id                 1537 non-null   int64         
 1   release_date       1537 non-null   datetime64[ns]
 2   movie              1537 non-null   object        
 3   production_budget  1537 non-null   float64       
 4   domestic_gross     1537 non-null   object        
 5   worldwide_gross    1537 non-null   float64       
 6   year               1537 non-null   int64         
 7   genres             1537 non-null   object        
 8   CPI                1537 non-null   float64       
 9   CPI_mult           1537 non-null   float64       
 10  gross2022          1537 non-null   float64       
 11  budget2022         1537 non-null   float64       
 12  net2022            1537 non-null   float64       
 13  roi                1537 non-null   float64       
 14  budget2022bin      1537 non-null   category      
dtypes: category(1), datetime64[ns](1), float64(8), int64(2), object(3)
memory usage: 182.0+ KB
```

Info table

Using the provided movie info data from Rotten Tomatoes we do the following:

- Remove blank box_office sales and non-dollar box office numbers
- Cleanup data types for dates, $ numbers and runtime
- Create year, month & season columns
- Join on CPI data from MN Fed, and adjust the dollar figure columns in new 2022 tagged columns

In [96]:
```
info
```

Out[96]:

	id	synopsis	rating	genre	director	writer	t
0	1	This gritty, fast-paced, and innovative police...	R	Action and Adventure|Classics|Drama	William Friedkin	Ernest Tidyman	
1	3	New York City, not-too-distant-future: Eric Pa...	R	Drama|Science Fiction and Fantasy	David Cronenberg	David Cronenberg|Don DeLillo	
2	5	Illeana Douglas delivers a superb performance	R	Drama|Musical and Performing Arts	Allison Anders	Allison Anders	

		...					
3	6	Michael Douglas runs afoul of a treacherous su...	R	Drama|Mystery and Suspense	Barry Levinson	Paul Attanasio|Michael Crichton	
4	7	NaN	NR	Drama|Romance	Rodney Bennett	Giles Cooper	
...	...	...	...	...	...	...	
1555	1996	Forget terrorists or hijackers -- there's a ha...	R	Action and Adventure|Horror|Mystery and Suspense	NaN	NaN	A
1556	1997	The popular Saturday Night Live sketch was exp...	PG	Comedy|Science Fiction and Fantasy	Steve Barron	Terry Turner|Tom Davis|Dan Aykroyd|Bonnie Turner	
1557	1998	Based on a novel by Richard Powell, when the l...	G	Classics|Comedy|Drama|Musical and Performing Arts	Gordon Douglas	NaN	
1558	1999	The Sandlot is a coming-of-age story about a g...	PG	Comedy|Drama|Kids and Family|Sports and Fitness	David Mickey Evans	David Mickey Evans|Robert Gunter	
1559	2000	Suspended from the force, Paris cop Hubert is ...	R	Action and Adventure|Art House and Internation...	NaN	Luc Besson	

1560 rows × 12 columns

In [97]:
```
# cleanup info table to just non-null box office sales
# has genre, director, genre, runtime
clean_info = info.loc[(~info['box_office'].isnull()) & (info['currency']=='$'
clean_info['theater_date'] = pd.to_datetime(clean_info['theater_date'])
clean_info['year'] = clean_info['theater_date'].dt.year
clean_info['month'] = clean_info['theater_date'].dt.month

# join CPI info & inflate $ to 2022 figures
clean_info = clean_info.merge(cpi, on='year', how='left')
clean_info["box_office"] = clean_info["box_office"].replace("[$,]", "", regex
clean_info['box2022'] = clean_info['box_office'] * clean_info['CPI_mult']
clean_info['runtime'] = clean_info['runtime'].replace("[minutes]", "", regex="

# map months to seasons
m2s = {1.0: "Winter", 2.0: "Winter", 3.0: "Spring", 4.0: "Spring", 5.0: "Spri
clean_info['season'] = clean_info['month'].map(m2s)
clean_info
```

Out[97]:

	id	synopsis	rating	genre	director	writer	theate
0	3	New York City, not-too-distant-future: Eric Pa...	R	Drama|Science Fiction and Fantasy	David Cronenberg	David Cronenberg|Don DeLillo	2012
1	10	Some cast and crew from NBC's highly acclaimed...	PG-13	Comedy	Jake Kasdan	Mike White	200
2	13	Stewart Kane, an Irishman living in the Austra...	R	Drama	Ray Lawrence	Raymond Carver|Beatrix Christian	2006
3	14	"Love Ranch" is a bittersweet love story that ...	R	Drama	Taylor Hackford	Mark Jacobson	2010
4	22	Two-time Academy Award Winner Kevin Spacey giv...	R	Comedy|Drama|Mystery and Suspense	George Hickenlooper	Norman Snider	201(
...	...	...	...	...	...	...	
		A band of renegades		Action and			

335	1980	on the run in outer space ...	PG-13	Adventure|Science Fiction and Fantasy	Joss Whedon	Joss Whedon	2005
336	1981	Money, Fame and the Knowledge of English. In I...	NR	Comedy|Drama	Gauri Shinde	Gauri Shinde	2012
337	1985	A woman who joins the undead against her will ...	R	Horror|Mystery and Suspense	Sebastian Gutierrez	Sebastian Gutierrez	2007
338	1986	Aki Kaurismaki's The Man Without a Past opens ...	PG	Art House and International|Comedy|Drama	NaN	NaN	2002
339	1996	Forget terrorists or hijackers -- there's a ha...	R	Action and Adventure|Horror|Mystery and Suspense	NaN	NaN	2006

340 rows × 18 columns

Clean info check & remove dupes

In [109…
```
clean_info.describe()
```

Out[109…

	id	box_office	runtime	year	month	CPI	CPI
count	340.000000	3.400000e+02	338.000000	334.000000	334.000000	334.000000	334.00
mean	1026.523529	3.790601e+07	106.698225	2007.859281	6.943114	208.564072	1.46
std	577.879413	5.749159e+07	19.621131	6.165969	3.434465	26.756044	0.56
min	3.000000	3.630000e+02	15.000000	1958.000000	1.000000	28.900000	1.1
25%	504.750000	1.905152e+06	93.000000	2004.000000	4.000000	188.900000	1.28
50%	1074.000000	1.414105e+07	105.000000	2008.000000	7.000000	214.500000	1.37
75%	1525.500000	4.482524e+07	118.000000	2012.000000	10.000000	229.600000	1.55
max	1996.000000	3.680000e+08	229.000000	2018.000000	12.000000	251.100000	10.18

In [110…
```
clean_info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 340 entries, 0 to 339
Data columns (total 19 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   id            340 non-null    int64         
 1   synopsis      340 non-null    object        
 2   rating        340 non-null    object        
 3   genre         340 non-null    object        
 4   director      299 non-null    object        
 5   writer        273 non-null    object        
 6   theater_date  334 non-null    datetime64[ns]
 7   dvd_date      334 non-null    object        
 8   currency      340 non-null    object        
 9   box_office    340 non-null    float64       
 10  runtime       338 non-null    float64       
 11  studio        305 non-null    object        
 12  year          334 non-null    float64       
 13  month         334 non-null    float64       
 14  CPI           334 non-null    float64       
 15  CPI_mult      334 non-null    float64       
 16  box2022       334 non-null    float64       
 17  season        334 non-null    object        
 18  monthName     334 non-null    object        
dtypes: datetime64[ns](1), float64(7), int64(1), object(10)
memory usage: 53.1+ KB
```

In [117…
```
# remove dupes, removes 0 rows
clean_info = clean_info.drop_duplicates()
clean_info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 340 entries, 0 to 339
Data columns (total 19 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   id             340 non-null    int64         
 1   synopsis       340 non-null    object        
 2   rating         340 non-null    object        
 3   genre          340 non-null    object        
 4   director       299 non-null    object        
 5   writer         273 non-null    object        
 6   theater_date   334 non-null    datetime64[ns]
 7   dvd_date       334 non-null    object        
 8   currency       340 non-null    object        
 9   box_office     340 non-null    float64       
 10  runtime        338 non-null    float64       
 11  studio         305 non-null    object        
 12  year           334 non-null    float64       
 13  month          334 non-null    float64       
 14  CPI            334 non-null    float64       
 15  CPI_mult       334 non-null    float64       
 16  box2022        334 non-null    float64       
 17  season         334 non-null    object        
 18  monthName      334 non-null    object        
dtypes: datetime64[ns](1), float64(7), int64(1), object(10)
memory usage: 53.1+ KB
```

Genre cleanup

Before analyzing by genre, we need to expand out the genre column to its consituent genres.

The movies are tagged with all the applicable genres that pertain to the film, but we want to analyze each genre.

In [118…

```
# explode out multi-genre films to each of their component genres
g2n = clean_budgets[['genres','net2022']]
g2n['genres']=g2n['genres'].apply(lambda x : x.split(','))
g2n = g2n.join(pd.concat([g2n.pop('genres').explode()],axis=1))
```

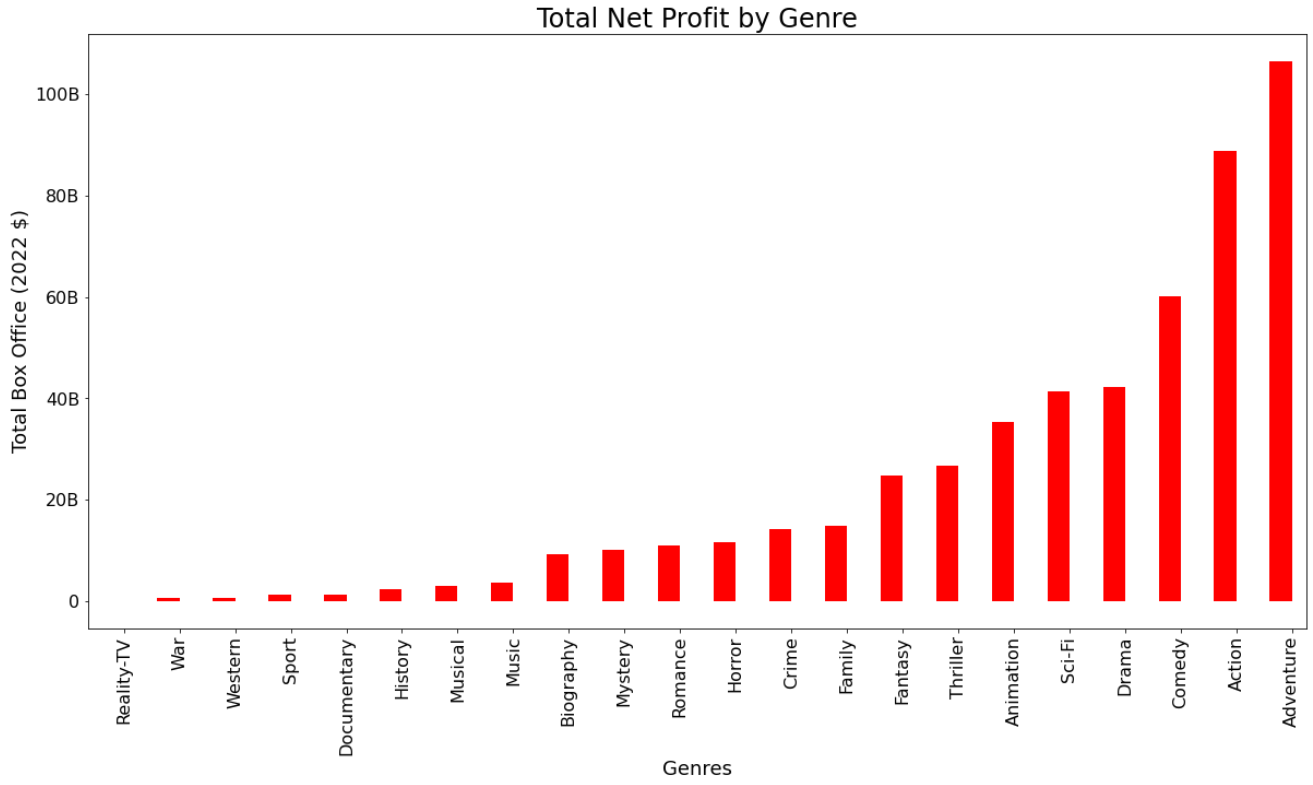
In [119…

```
# helper function for axis formatting of large numbers

def numformat(num, pos):
    mag = 0
    while abs(num) >= 1000:
        mag += 1
        num /= 1000.0
    # add more suffixes if you need them
    return '%.0f%s' % (num, ['', 'K', 'M', 'B', 'T'][mag])
```

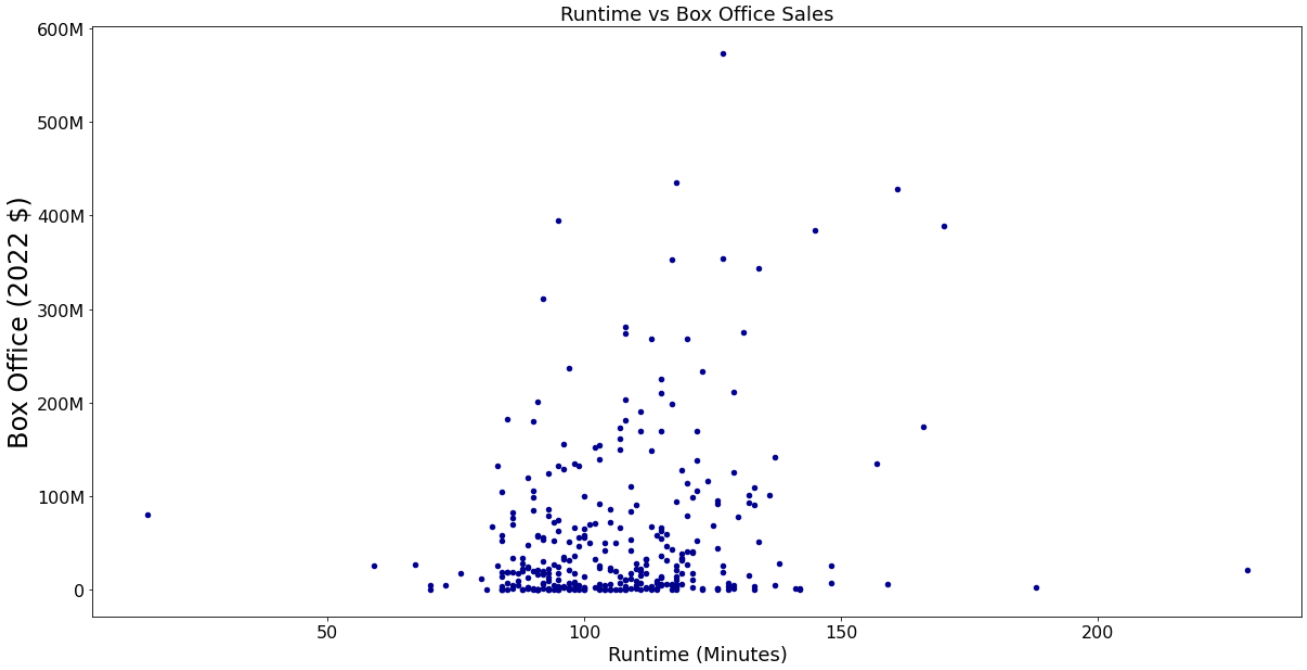
Genre vs Profit visualizations

- Immediately this bar chart makes a lot of sense and has clear takeaways

In [120…
```
# visualize total box office sales by genre
tot_net = g2n[['genres','net2022']].groupby('genres').sum()
tot_net = tot_net.sort_values(by=['net2022'])
plt.rcParams['figure.figsize'] = (20, 10)
fig = plt.figure() # Create matplotlib figure
ax0 = fig.add_subplot(111) # Create matplotlib axes
width = 0.4
tot_net.net2022.plot(kind='bar', color='red', ax=ax0, width=width, position=1

formatter = FuncFormatter(numformat)
ax0.yaxis.set_major_formatter(formatter)

ax0.tick_params(axis='both', which='major', labelsize=16)
ax0.set_title('Total Net Profit by Genre', fontsize=24)
ax0.set_xlabel('Genres', fontsize=18)
ax0.set_ylabel('Total Box Office (2022 $)', fontsize=18)
plt.show()
```



In [101…
```
ax1 = clean_info.plot.scatter(x='runtime',y='box2022',c='DarkBlue')
ax1.tick_params(axis='both', which='major', labelsize=16)
ax1.set_title('Runtime vs Box Office Sales', fontsize=18)
ax1.set_xlabel('Runtime (Minutes)', fontsize=18)
ax1.set_ylabel('Box Office (2022 $)', fontsize=24)

formatter = FuncFormatter(numformat)
ax1.yaxis.set_major_formatter(formatter)
plt.show()
```

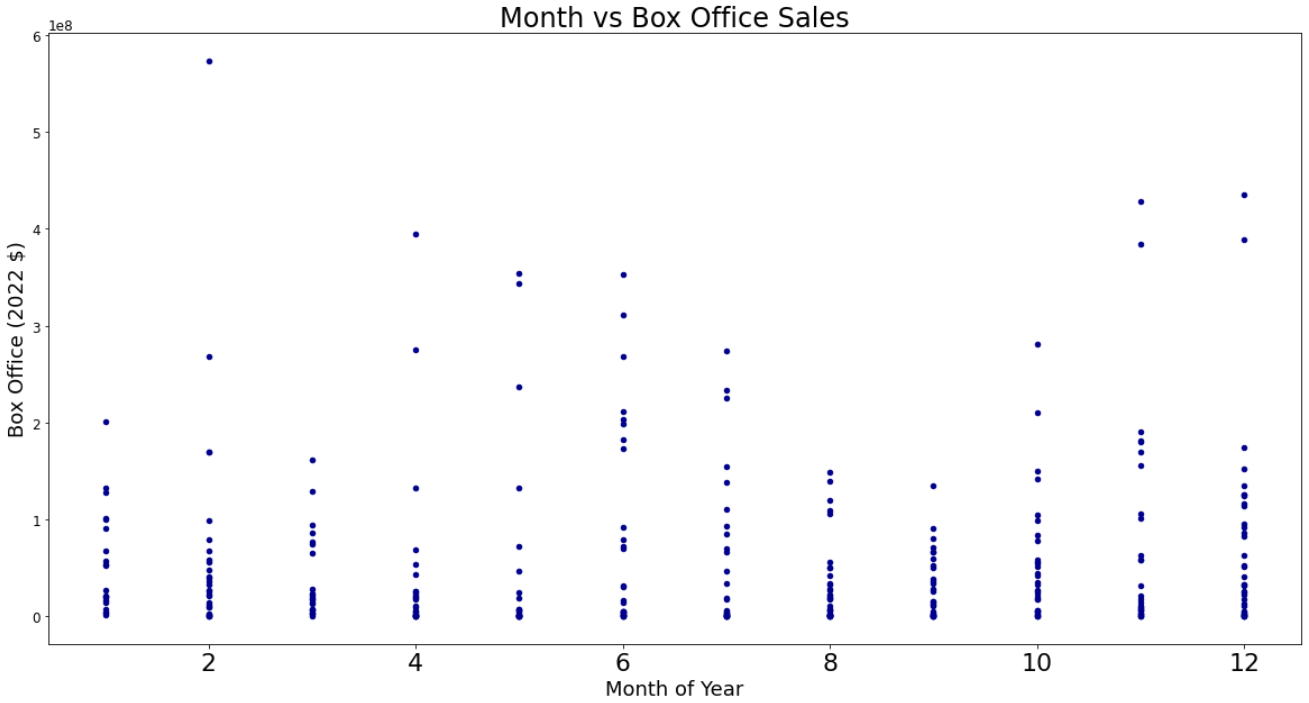


Release date visualizations

- First we try a scatter of Month vs Box Office Sales.. hard to read!
- Next we try a bar chart for Month vs Box Office Sales, interesting but noisy
- Finally we settle on mapping the months into seasons and visualizing Season vs Box Office Sales - nice

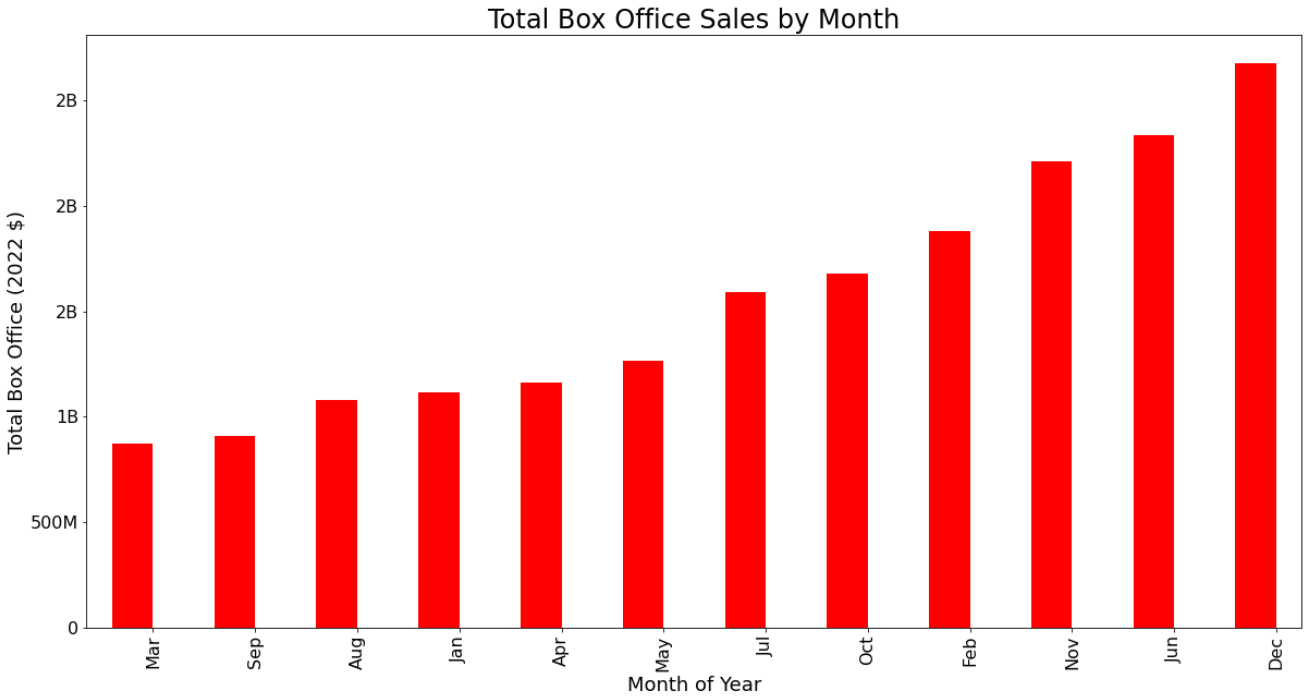
In [102…

```
# Try Month vs Sales scatter
ax2 = clean_info.plot.scatter(x='month',y='box2022',c='DarkBlue')
ax2.set_title('Month vs Box Office Sales', fontsize=24)
ax2.set_xlabel('Month of Year', fontsize=18)
ax2.set_ylabel('Box Office (2022 $)', fontsize=18)
plt.show()
```



In [103…

```
# Try Total Box office by Month of Year
# visualize total box office sales by genre
num2mo = {1.0:'Jan', 2.0:'Feb', 3.0:'Mar', 4.0:'Apr', 5.0:'May', 6.0:'Jun', 7
clean_info['monthName'] = clean_info['month'].map(num2mo)
tot_mo = clean_info[['monthName','box2022']].groupby('monthName').sum()
tot_mo = tot_mo.sort_values(by=['box2022'])
# map months to seasons
# tot_mo['monthName'] = tot_mo['month'].map(num2mo)

plt.rcParams['figure.figsize'] = (20, 10)
fig = plt.figure() # Create matplotlib figure
ax3 = fig.add_subplot() # Create matplotlib axes
tot_mo.box2022.plot(kind='bar', color='red', ax=ax3, width=width, position=1)
ax3.tick_params(axis='both', which='major', labelsize=16)
ax3.set_title('Total Box Office Sales by Month', fontsize=24)
ax3.set_xlabel('Month of Year', fontsize=18)
ax3.set_ylabel('Total Box Office (2022 $)', fontsize=18)

formatter = FuncFormatter(numformat)
ax3.yaxis.set_major_formatter(formatter)


plt.show()
tot_mo
```

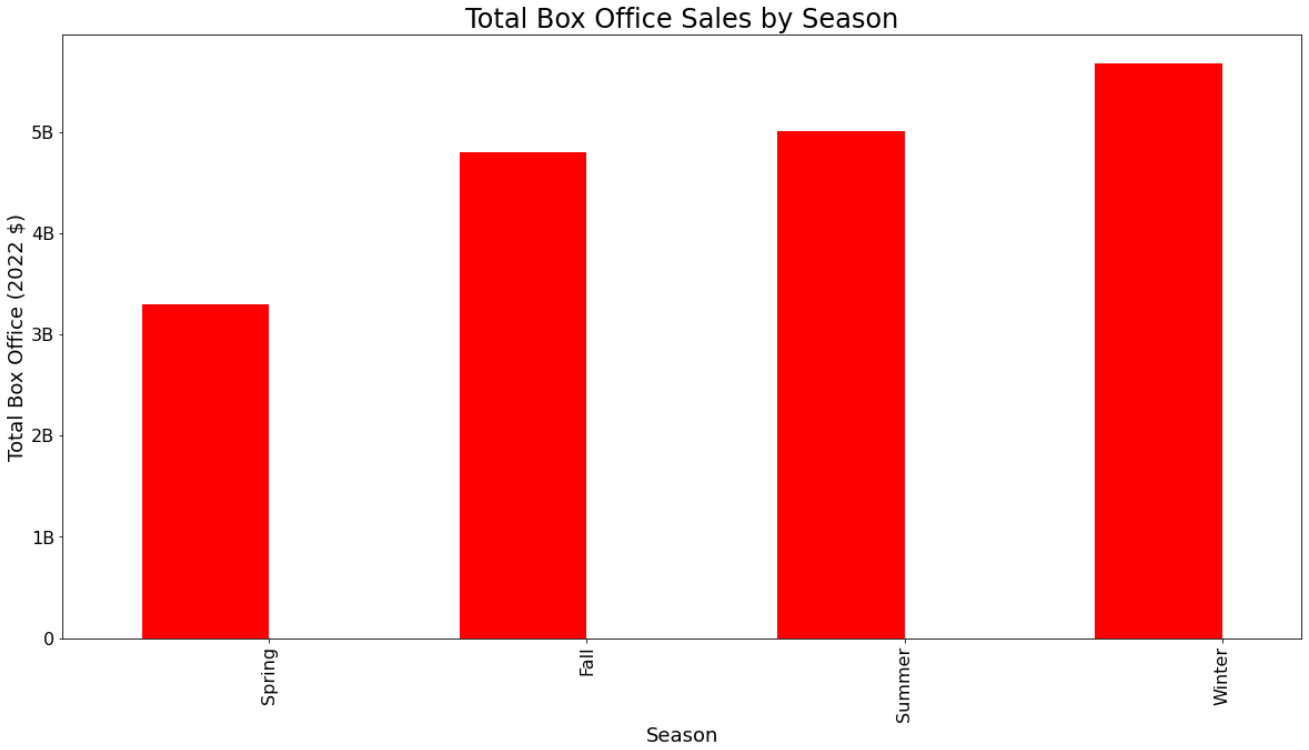


Out[103...

monthName	box2022
Mar	8.706781e+08
Sep	9.085333e+08
Aug	1.080705e+09
Jan	1.117243e+09
Apr	1.159247e+09
May	1.263007e+09
Jul	1.589970e+09
Oct	1.677649e+09
Feb	1.881460e+09
Nov	2.212100e+09
Jun	2.336928e+09
Dec	2.677792e+09

In [104…

```
# Try Total Box office by Season of Year
# visualize total box office sales by genre
tot_season = clean_info[['season','box2022']].groupby('season').sum()
tot_season = tot_season.sort_values(by=['box2022'])
plt.rcParams['figure.figsize'] = (20, 10)
fig = plt.figure() # Create matplotlib figure
ax4 = fig.add_subplot() # Create matplotlib axes
tot_season.box2022.plot(kind='bar', color='red', ax=ax4, width=width, positio
ax4.set_title('Total Box Office Sales by Season', fontsize=24)
ax4.tick_params(axis='both', which='major', labelsize=16)
ax4.set_xlabel('Season', fontsize=18)
ax4.set_ylabel('Total Box Office (2022 $)', fontsize=18)

formatter = FuncFormatter(numformat)
ax4.yaxis.set_major_formatter(formatter)
plt.show()
```

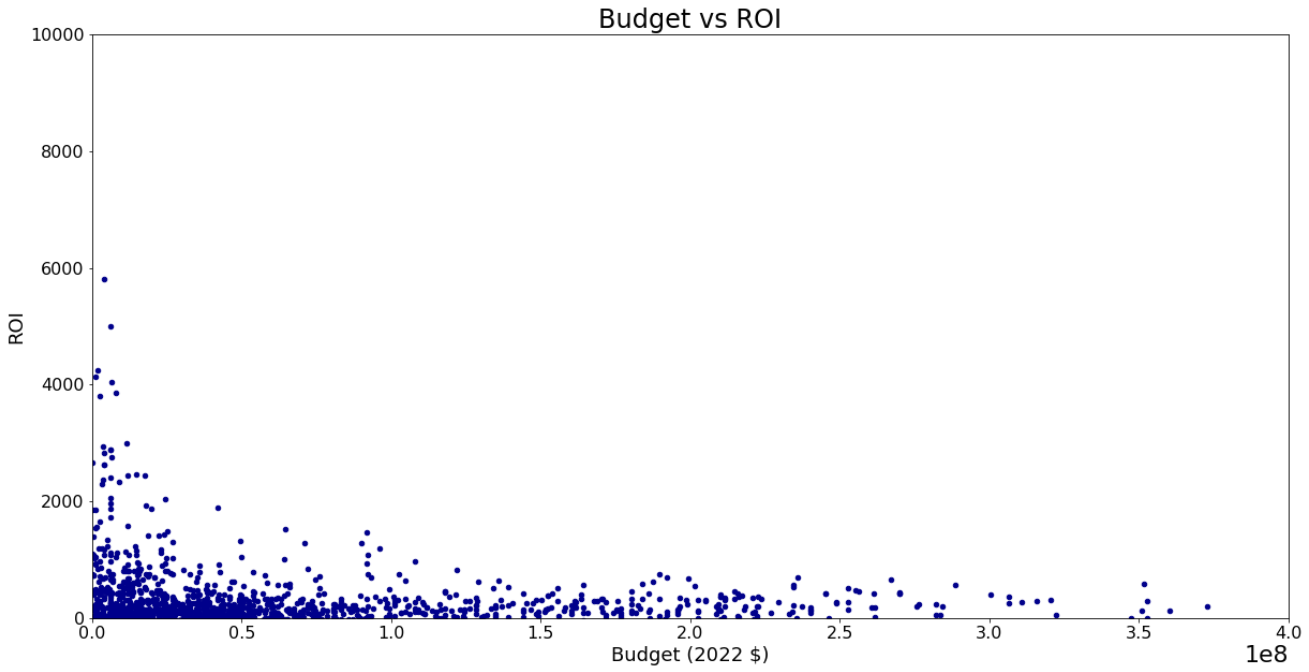


Budget vs ROI visualization

- First we try a scatter plot, and find it a bit hard to read
- Next we try bucketing the budgets and doing a bar chart - much better!

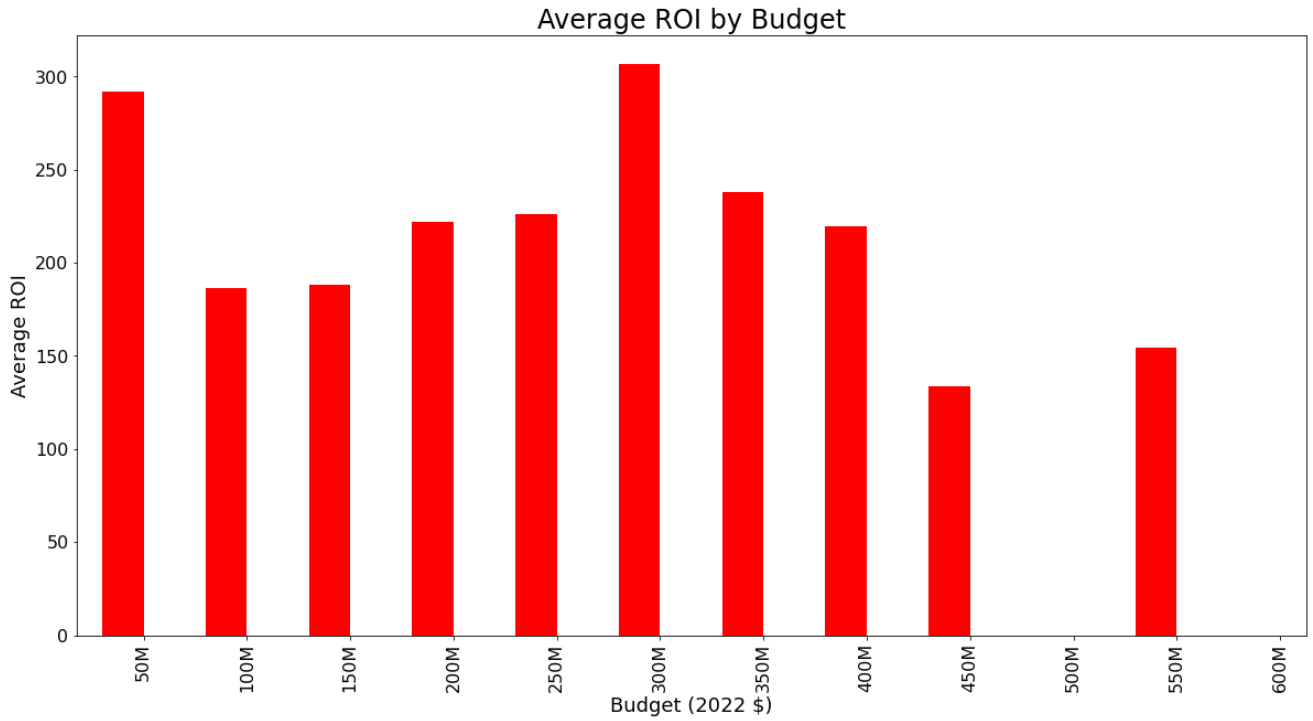
In [121…

```
#budget2022 vs #net2022

# clean_budgets
ax5 = clean_budgets.plot.scatter(x='budget2022',y='roi',c='DarkBlue')
plt.xlim(0, 4e8)
plt.ylim(0, 10000)
ax5.tick_params(axis='both', which='major', labelsize=16)
ax5.set_title('Budget vs ROI', fontsize=24)
ax5.set_xlabel('Budget (2022 $)', fontsize=18)
ax5.set_ylabel('ROI', fontsize=18)

plt.show()
```



In [122…

```
# bucketed ROI by budget graph
avg_roi = clean_budgets[['budget2022bin','roi']].groupby('budget2022bin').mea
plt.rcParams['figure.figsize'] = (20, 10)
fig = plt.figure() # Create matplotlib figure
ax6 = fig.add_subplot() # Create matplotlib axes
avg_roi.roi.plot(kind='bar', color='red', ax=ax6, width=width, position=1)
ax6.tick_params(axis='both', which='major', labelsize=16)
ax6.set_title('Average ROI by Budget', fontsize=24)
ax6.set_xlabel('Budget (2022 $)', fontsize=18)
ax6.set_ylabel('Average ROI', fontsize=18)



plt.show()
```

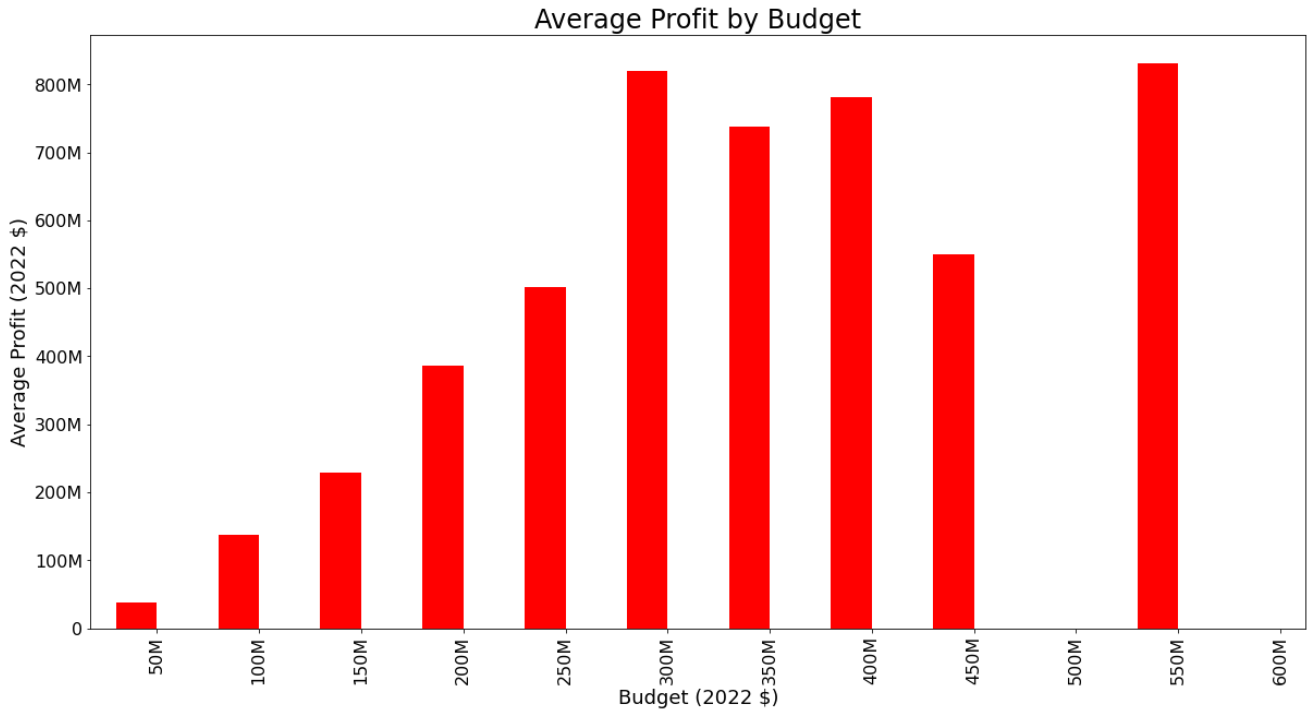


Budget vs Profit visualization

- The bucketed budget vs ROI bar chart makes sense, so lets use the same to look at Budget vs Profit
- This makes the relationship of money in to money out even more clear

In [123…

```
# bucketed ROI by budget graph
avg_net = clean_budgets[['budget2022bin','net2022']].groupby('budget2022bin')
plt.rcParams['figure.figsize'] = (20, 10)
fig = plt.figure() # Create matplotlib figure
ax7 = fig.add_subplot() # Create matplotlib axes
avg_net.net2022.plot(kind='bar', color='red', ax=ax7, width=width, position=1
ax7.tick_params(axis='both', which='major', labelsize=16)
ax7.set_title('Average Profit by Budget', fontsize=24)
ax7.set_xlabel('Budget (2022 $)', fontsize=18)
ax7.set_ylabel('Average Profit (2022 $)', fontsize=18)


formatter = FuncFormatter(numformat)
ax7.yaxis.set_major_formatter(formatter)

plt.show()
```



Recommendations & Next Steps

Recommendation

- Focus your studio on Action & Adventure movies as these genres are the most profitable
- Release films primarily in Winter & Summer seasons, as box office sales show moviegoing is highest in these seasons
- Budget up to 300M per film, as ROI and Profit increases with budget until 300M at which point it drops off
- Keep movie runtime in the sweet spot of 90-120 minutes length as higher run times do not result in higher sales

Next Steps

Multi-factor drill downs such as:

- Optimal budget for Action & Adventure genres individually
- Drill into day-of-week and week-of-year time slices
- Look at how time of year & budget interact
- Etc

In [108…

```
from IPython.display import HTML
from IPython.display import IFrame

IFrame(src="https://www.youtube.com/embed/b9434BoGkNQ", width="560", height="
```

Out[108…

In [ ]: