

## Final Project Submission

- Student name: Lucy Chepkemai Ruto
- Student pace: Part-time
- Scheduled project review date/time: 5th November 2023 at 8:00 P.M
- Instructor name: Samwel Jane
- Blog post URL:

## Microsoft Corporation movie

### Overview

This project explores the types of films that are currently doing the well at the box office and needs of the [Microsoft Corporation](https://www.microsoft.com/) (<https://www.microsoft.com/>), which wants to set-up a new movie studio. The Microsoft corporation can use these insights to decide what type of films to create.

### Business Problem

The Austin Animal Shelter may be able to improve their resource allocation to both reduce costs and ensure that the center has staff and space to care for the animals brought to them. Doing so will allow the Austin Animal Shelter to better serve its clients while also freeing up resources to expand the scope of services they can offer.



```
In [1]: ▶ # Importing Standard packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import prep as p
import datetime
```

```
In [2]: ▶ #Loading datasets required for analysis
title_basics = pd.read_csv('./ZippedData/title.basics.csv')
movie_budgets = pd.read_csv('./ZippedData/tn.movie_budgets.csv')
tmdb_movies = pd.read_csv('./ZippedData/tmdb.movies.csv')
```

```
In [3]: ▶ movie_budgets.head(3)
```

Out[3]:

	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

In [4]: `movie_budgets.shape`

Out[4]: (5782, 6)

In [5]: `movie_budgets.info()`

```
<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
```

```
In [6]: movie_budgets.drop(columns = ['id'], axis=1, inplace=True)
for column in ['production_budget', 'domestic_gross', 'worldwide_gross']:
    movie_budgets[column] = movie_budgets[column].str.replace('$', '', regex=False)
    movie_budgets[column] = movie_budgets[column].str.replace(',', '', regex=False)
    movie_budgets[column] = movie_budgets[column].astype('int64')
movie_budgets.head(10)
```

Out[6]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	Dec 18, 2009	Avatar	425000000	760507625	2776345279
1	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875
2	Jun 7, 2019	Dark Phoenix	350000000	42762350	149762350
3	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	1403013963
4	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1316721747
5	Dec 18, 2015	Star Wars Ep. VII: The Force Awakens	306000000	936662225	2053311220
6	Apr 27, 2018	Avengers: Infinity War	300000000	678815482	2048134200
7	May 24, 2007	Pirates of the Caribbean: At World's End	300000000	309420425	963420425
8	Nov 17, 2017	Justice League	300000000	229024295	655945209
9	Nov 6, 2015	Spectre	300000000	200074175	879620923

```
In [7]: movie_budgets['profit'] = movie_budgets['worldwide_gross'] - movie_budgets['production_budget']
movie_budgets['ROI'] = movie_budgets['profit'] / movie_budgets['production_budget']
movie_budgets['profit_status'] = np.where(movie_budgets['ROI'] > 0, "Profit", "Loss")
movie_budgets['profitable_oversees'] = movie_budgets['worldwide_gross'] - movie_budgets['domestic_gross']
movie_budgets.sample(10)
```

Out[7]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	profit	ROI	profit_status	profitable_oversees
514	Jan 20, 2017	xXx: Return of Xander Cage	85000000	44898413	345033359	260033359	3.059216	Profit	300134946
4802	May 2, 2014	Ida	2600000	3827060	15298355	12698355	4.883983	Profit	11471295
1915	Oct 19, 2007	30 Days of Night	30000000	39568996	80276156	50276156	1.675872	Profit	40707160
4386	Feb 17, 2009	Dead Like Me: Life After Death	5000000	0	0	-5000000	-1.000000	Loss	0
2723	Jul 3, 1985	Back to the Future	19000000	212259762	385524784	366524784	19.290778	Profit	173265022
1020	Dec 25, 1996	Evita	55000000	50047179	151947179	96947179	1.762676	Profit	101900000
2023	Dec 5, 2008	Frost/Nixon	29000000	18622031	28452945	-547055	-0.018864	Loss	9830914
1768	Dec 4, 1981	Reds	33500000	50000000	50000000	16500000	0.492537	Profit	0
3076	Jun 26, 2009	The Hurt Locker	15000000	17017811	49894223	34894223	2.326282	Profit	32876412
1431	Jan 25, 2002	The Count of Monte Cristo	40000000	54228104	75389090	35389090	0.884727	Profit	21160986

```
In [8]: duplicates = movie_budgets[movie_budgets[['movie', 'release_date']].duplicated()]
```

```
In [9]: duplicates
```

Out[9]:

release_date	movie	production_budget	domestic_gross	worldwide_gross	profit	ROI	profit_status	profitable_oversees
--------------	-------	-------------------	----------------	-----------------	--------	-----	---------------	---------------------

In [10]: ► movie\_budgets.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5782 entries, 0 to 5781
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   release_date          5782 non-null   object
1   movie                 5782 non-null   object
2   production_budget     5782 non-null   int64
3   domestic_gross        5782 non-null   int64
4   worldwide_gross       5782 non-null   int64
5   profit               5782 non-null   int64
6   ROI                  5782 non-null   float64
7   profit_status         5782 non-null   object
8   profitable_oversees    5782 non-null   int64
dtypes: float64(1), int64(5), object(3)
memory usage: 406.7+ KB
```

In [11]: ► movie\_budgets['release\_date'] = pd.to\_datetime(movie\_budgets['release\_date'])

```
In [12]: ► movie_budgets['release_day_of_week'] = movie_budgets['release_date'].dt.day_name()  
movie_budgets.sample(30)
```

Out[12]:

	release_date	movie	production_budget	domestic_gross	worldwide_gross	profit	ROI	profit_status	profitable_oversees	release
<b>1108</b>	2017-12-22	The Post	50000000	81903458	179748880	129748880	2.594978	Profit	97845422	
<b>280</b>	2017-01-13	Monster Trucks	125000000	33370166	61642798	-63357202	-0.506858	Loss	28272632	
<b>2671</b>	2005-01-28	Alone in the Dark	20000000	5178569	10588079	-9411921	-0.470596	Loss	5409510	
<b>2724</b>	1990-11-09	Dances with Wolves	19000000	184208842	424200000	405200000	21.326316	Profit	239991158	
<b>771</b>	2000-12-15	What Women Want	65000000	182805123	374105123	309105123	4.755463	Profit	191300000	
<b>1109</b>	1999-02-05	Payback	50000000	81526121	161626121	111626121	2.232522	Profit	80100000	
<b>3908</b>	2002-09-20	8 femmes	8000000	3076425	42376425	34376425	4.297053	Profit	39300000	
<b>1988</b>	1984-06-01	Once Upon a Time in America	30000000	5321508	5575648	-24424352	-0.814145	Loss	254140	
<b>3883</b>	1982-11-10	Creepshow	8000000	20036244	20036244	12036244	1.504530	Profit	0	
<b>2293</b>	2009-11-25	The Road	25000000	8114270	29206732	4206732	0.168269	Profit	21092462	
<b>3909</b>	1991-01-01	Showdown in Little Tokyo	8000000	2275557	2275557	-5724443	-0.715555	Loss	0	
<b>253</b>	2008-11-26	Australia	130000000	49554002	215080810	85080810	0.654468	Profit	165526808	
<b>2060</b>	2000-12-22	Dracula 2000	28000000	33000377	33000377	5000377	0.178585	Profit	0	
<b>2972</b>	1997-08-15	Steel	16000000	1686429	1686429	-14313571	-0.894598	Loss	0	
<b>171</b>	2009-05-01	X-Men Origins: Wolverine	150000000	179883157	374825760	224825760	1.498838	Profit	194942603	
<b>999</b>	2002-12-13	Maid in Manhattan	55000000	93932896	163838217	108838217	1.978877	Profit	69905321	
<b>2156</b>	1996-05-17	Flipper	25530000	20080020	30593313	5063313	0.198328	Profit	10513293	
<b>3648</b>	2006-12-01	Turistas	10000000	7027762	14321070	4321070	0.432107	Profit	7293308	
<b>3041</b>	2000-04-28	Where the Heart Is	15000000	33771174	40862054	25862054	1.724137	Profit	7090880	
<b>5766</b>	2006-04-28	Clean	10000	138711	138711	128711	12.871100	Profit	0	

	release_date	movie	production_budget	domestic_gross	worldwide_gross	profit	ROI	profit_status	profitable_oversees	release
5519	2015-03-31	Viskningar och rop	400000	0	9071	-390929	-0.977322	Loss	9071	
369	2010-12-22	Little Fockers	100000000	148438600	310650574	210650574	2.106506	Profit	162211974	
3308	2000-12-29	The Claim	13000000	622023	1375635	-11624365	-0.894182	Loss	753612	
4803	1987-01-01	Maurice	2600000	3147950	3198308	598308	0.230118	Profit	50358	
4560	2008-08-26	Purple Violets	4000000	0	0	-4000000	-1.000000	Loss	0	
2021	2016-08-12	Florence Foster Jenkins	29000000	27383770	56000339	27000339	0.931046	Profit	28616569	
5375	2007-04-13	Aqua Teen Hunger Force Colon Movie Film for Theaters	750000	5520368	5520368	4770368	6.360491	Profit	0	
3147	1996-12-31	Silent Trigger	15000000	76382	76382	-14923618	-0.994908	Loss	0	
3642	2003-07-18	Dirty Pretty Things	10000000	8112414	14156753	4156753	0.415675	Profit	6044339	
1853	2004-12-15	Million Dollar Baby	30000000	100492203	231928227	201928227	6.730941	Profit	131436024	

In [13]: `movie_budgets = movie_budgets[['movie', 'release_date', 'release_day_of_week', 'worldwide_gross', 'production_budget', 'profit', 'ROI', 'profit_status', 'profitable_oversees', 'release_date']]`

In [14]: `movie_budgets.sort_values(by="profit", axis=0, ascending=False, inplace=True)`



In [15]: ▶

movie\_budgets.head(100)

Out[15]:

movie	release_date	release_day_of_week	worldwide_gross	production_budget	profit	ROI	profit_status	domestic_gross	profitable_over
Avatar	2009-12-18	Friday	2776345279	425000000	2351345279	5.532577	Profit	760507625	201583
Titanic	1997-12-19	Friday	2208208395	200000000	2008208395	10.041042	Profit	659363944	154884
Avengers: Infinity War	2018-04-27	Friday	2048134200	300000000	1748134200	5.827114	Profit	678815482	136931
Star Wars Ep. VII: The Force Awakens	2015-12-18	Friday	2053311220	306000000	1747311220	5.710167	Profit	936662225	111664
Jurassic World	2015-06-12	Friday	1648854864	215000000	1433854864	6.669092	Profit	652270625	99658
...	...	...	...	...	...	...	...	...	...
The Passion of the Christ	2004-02-25	Wednesday	622341924	25000000	597341924	23.893677	Profit	370782930	25158
Spider-Man 2	2004-06-30	Wednesday	795110670	200000000	595110670	2.975553	Profit	373524485	42158
Shrek Forever After	2010-05-21	Friday	756244673	165000000	591244673	3.583301	Profit	238736787	51750
The Matrix Reloaded	2003-05-15	Thursday	738576929	150000000	588576929	3.923846	Profit	281553689	45702
Beauty and the Beast	1991-11-13	Wednesday	608431132	20000000	588431132	29.421557	Profit	376057266	23237

s × 10 columns

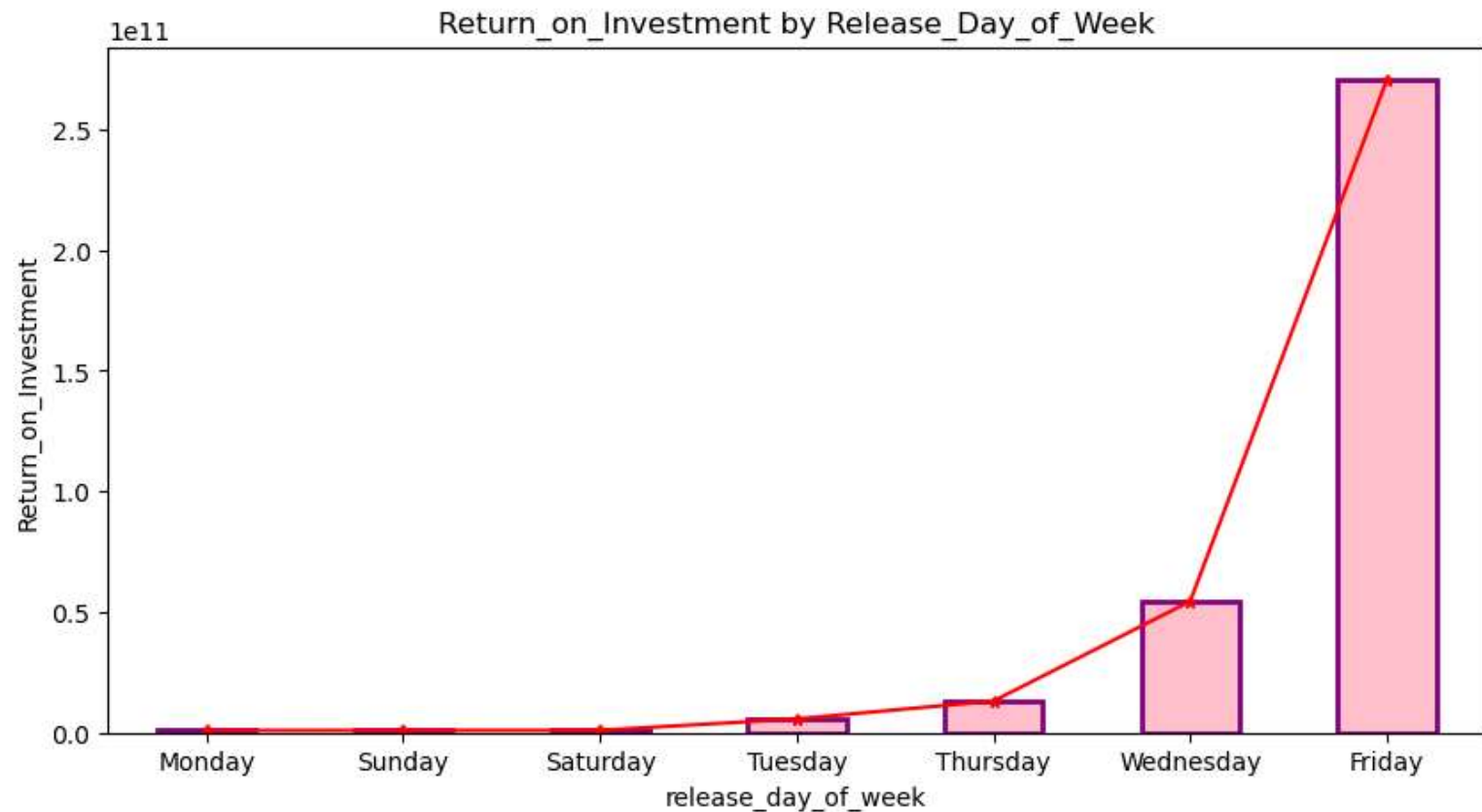


# Analysis

## Best day to realease a movie

```
In [16]: movie_budgets.groupby(['release_day_of_week'])['profit'].sum().sort_values().plot(kind='bar', figsize=(10,5),
        color="pink",edgecolor="purple",linewidth=2)
movie_budgets.groupby(['release_day_of_week'])['profit'].sum().sort_values().plot(kind='line', marker="*",
        color="red",ms=5)
plt.title('Return_on_Investment by Release_Day_of_Week')
plt.ylabel('Return_on_Investment')
```

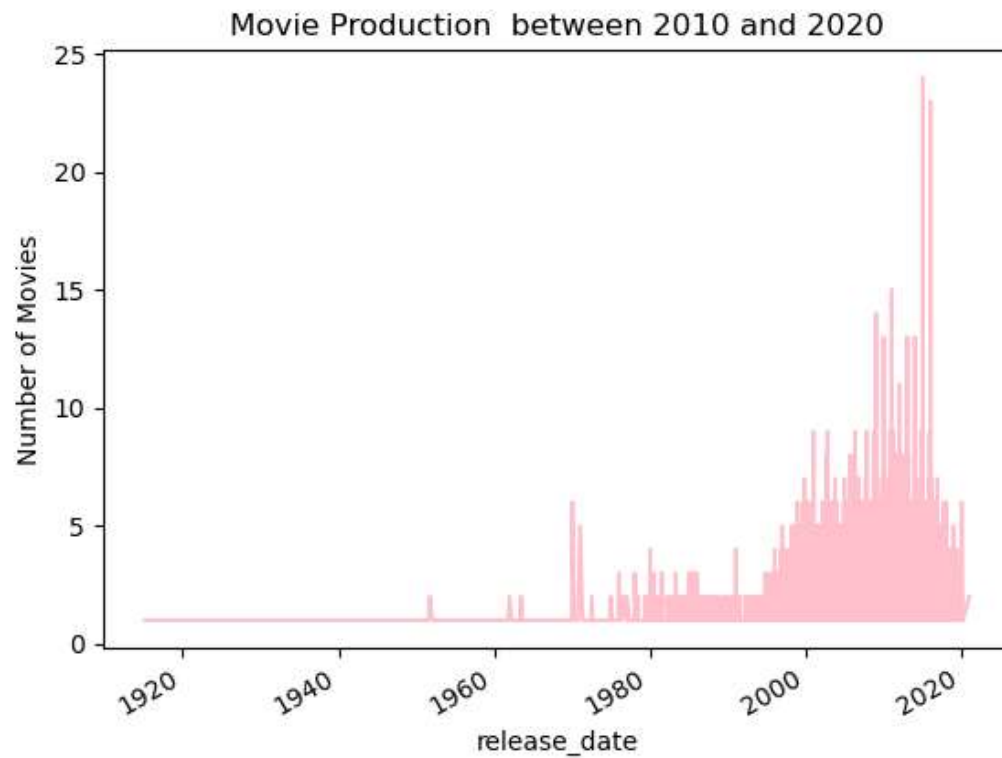
Out[16]: Text(0, 0.5, 'Return\_on\_Investment')



According to the analysis shown above, the best day to release a movie and receive the highest return on Investment is on Friday. The worst days would be saturday through Monday

```
In [159]: movie_budgets.groupby(['release_date'])['movie'].count().plot(kind='line',  
                                     color="pink",ms=10)  
plt.title('Movie Production between 2010 and 2020')  
plt.ylabel('Number of Movies')
```

Out[159]: Text(0, 0.5, 'Number of Movies')



```
In [17]: tmdb_movies.shape
```

Out[17]: (26517, 10)

In [18]: `tmdb_movies.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26517 entries, 0 to 26516
Data columns (total 10 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Unnamed: 0             26517 non-null  int64
1   genre_ids              26517 non-null  object
2   id                     26517 non-null  int64
3   original_language     26517 non-null  object
4   original_title         26517 non-null  object
5   popularity             26517 non-null  float64
6   release_date           26517 non-null  object
7   title                  26517 non-null  object
8   vote_average           26517 non-null  float64
9   vote_count             26517 non-null  int64
dtypes: float64(2), int64(3), object(5)
memory usage: 2.0+ MB
```

In [23]: `tmdb_movies.head(3)`

Out[23]:

	genre_ids	popularity	title	vote_average	vote_count
0	[12, 14, 10751]	33.533	Harry Potter and the Deathly Hallows: Part 1	7.7	10788
1	[14, 12, 16, 10751]	28.734	How to Train Your Dragon	7.7	7610
2	[12, 28, 878]	28.515	Iron Man 2	6.8	12368

In [26]: `tmdb_movies.columns`

Out[26]: Index(['genre\_ids', 'popularity', 'movie', 'vote\_average', 'vote\_count'], dtype='object')

In [21]: `tmdb_movies.drop(columns=['Unnamed: 0', 'original_language', 'original_title', 'release_date', 'id'], inplace=True, axis=1)`

In [25]: `tmdb_movies.rename(columns={"title": "movie"}, inplace=True)`

In [161]: `tmdb_movies_and_budgets = movie_budgets.set_index("movie").join(tmdb_movies.set_index("movie"), how="left")`

## Data Cleaning

For the intake and outcome files, I make them easier to work with by normalizing column names and dropping unnecessary columns.

```
In [162]: ▶ tmdb_movies_and_budgets.shape
```

```
Out[162]: (6190, 13)
```

```
In [163]: ▶ tmdb_movies_and_budgets.reset_index(inplace=True)
```

```
In [164]: ▶ duplicates = tmdb_movies_and_budgets.duplicated()
           duplicates.value_counts()
```

```
Out[164]: False    6016
           True     174
           dtype: int64
```

```
In [165]: ▶ title_basics.columns
```

```
Out[165]: Index(['tconst', 'movie', 'original_title', 'start_year', 'runtime_minutes',
                 'genres'],
                 dtype='object')
```

```
In [43]: ▶ title_basics.head(10)
```

```
Out[43]:
```

	tconst	movie	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
5	tt0111414	A Thin Life	A Thin Life	2018	75.0	Comedy
6	tt0112502	Bigfoot	Bigfoot	2017	NaN	Horror,Thriller
7	tt0137204	Joe Finds Grace	Joe Finds Grace	2017	83.0	Adventure,Animation,Comedy
8	tt0139613	O Silêncio	O Silêncio	2012	NaN	Documentary,History
9	tt0144449	Nema aviona za Zagreb	Nema aviona za Zagreb	2012	82.0	Biography

```
In [166]: ▶ title_basics.rename(columns={"primary_title": "movie"}, inplace=True)
```


```
In [167]: combined_dataset = tmdb_movies_and_budgets.set_index("movie").join(title_basics.set_index("movie"), how="left")
```

```
In [168]: duplicates = combined_dataset.duplicated()  
duplicates.value_counts()
```

```
Out[168]: False    8522  
          True     269  
          dtype: int64
```

```
In [169]: combined_dataset.reset_index(inplace=True)
```

```
In [170]: combined_dataset = combined_dataset.drop_duplicates(subset=['movie', 'release_date', 'production_budget'])
```

In [171]:  combined\_dataset.head(20)

Out[171]:

imdb_id	gross	profitable_oversees	genre_ids	popularity	vote_average	vote_count	tconst	original_title	start_year	runtime_minutes	genres
tt0073190	0	0	[18, 9648, 27, 53]	6.099	3.3	102.0	tt3526286	#Horror	2015.0	101.0	Crime,Drama,Thriller
tt0425665	2425665	2013395	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
tt082999	2082999	36203423	[53, 878, 18]	17.892	6.9	4629.0	tt1179933	10 Cloverfield Lane	2016.0	103.0	Drama,Horror,Mystery
tt04616	14616	0	[18]	0.955	5.4	7.0	tt3453052	10 Days in a Madhouse	2015.0	111.0	Drama
tt0177966	177966	22235984	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
tt0784201	784201	174281477	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
tt03941559	3941559	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
tt054702	54702	89080	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
tt0000000	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
tt0000000	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
tt0141459	141459	111700000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
tt0234694	234694	5071954	NaN	NaN	NaN	NaN	tt3517850	12 Rounds	2017.0	NaN	Action,Drama,Romance
tt0819713	819713	25298665	[10752, 18, 36, 28]	13.183	5.6	1312.0	tt1413492	12 Strong	2018.0	130.0	Action,Drama,History
tt0671993	671993	124353350	[18, 36]	16.493	7.9	6631.0	tt2024544	12 Years a Slave	2013.0	134.0	Biography,Drama,History
tt0335230	335230	41881941	[12, 18, 53]	11.435	7.0	4469.0	tt1542344	127 Hours	2010.0	94.0	Adventure,Biography,Drama
tt0139723	139723	40518989	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
tt0853219	853219	16558151	[28, 18, 36, 53, 10752]	21.486	7.0	1573.0	NaN	NaN	NaN	NaN	
tt09134	9134	38418	[27, 53]	10.899	6.3	576.0	tt2059171	13 Sins	2014.0	93.0	Horror,Thriller



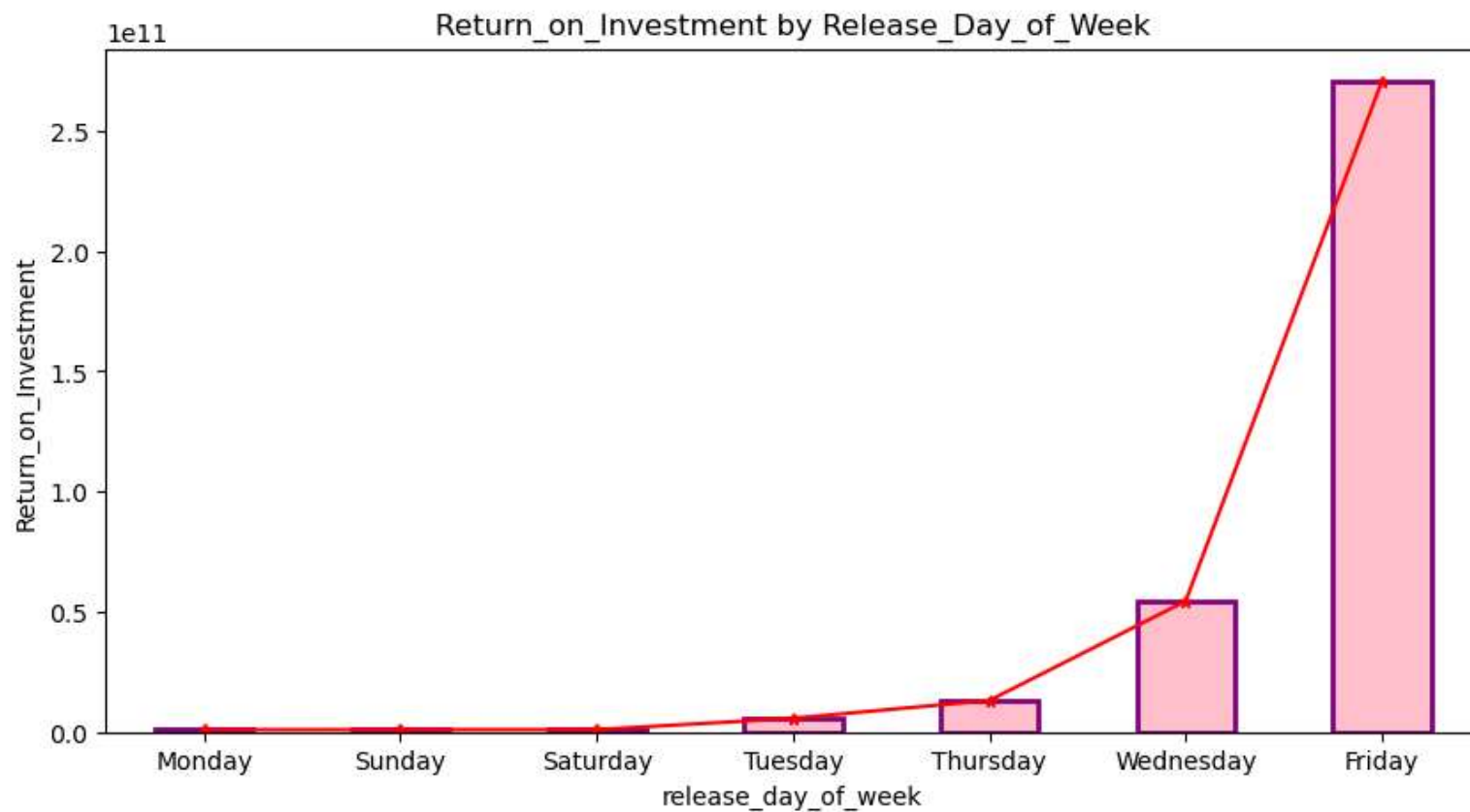
_gross	profitable_oversees	genre_ids	popularity	vote_average	vote_count	tconst	original_title	start_year	runtime_minutes	ge
1985628	59277742	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1375436	31956428	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

In [172]: `combined_dataset['start_year'].value_counts()`

```
Out[172]: 2011.0    311
          2010.0    304
          2014.0    290
          2013.0    273
          2012.0    267
          2015.0    264
          2016.0    221
          2017.0    167
          2018.0    163
          2019.0     99
          2020.0     13
          2021.0      4
          Name: start_year, dtype: int64
```

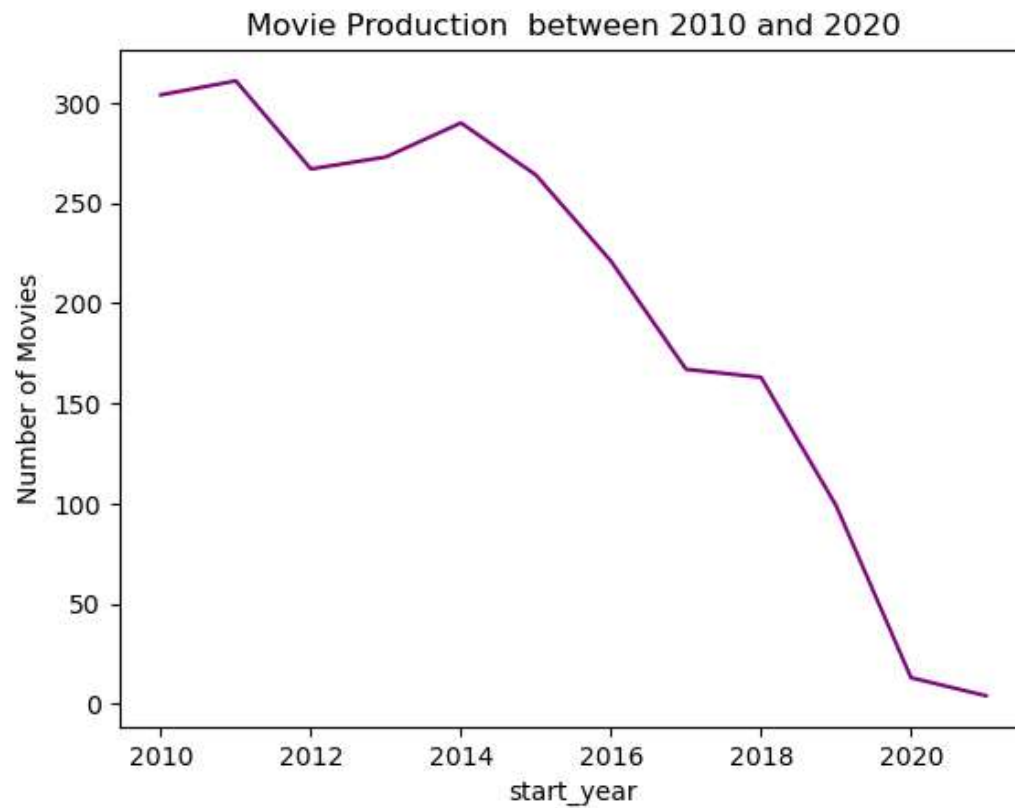
```
In [174]: combined_dataset.groupby(['release_day_of_week'])['profit'].sum().sort_values().plot(kind='bar', figsize=(10,5),
        color="pink",edgecolor="purple",linewidth=2)
combined_dataset.groupby(['release_day_of_week'])['profit'].sum().sort_values().plot(kind='line', marker="*",
        color="red",ms=5)
plt.title('Return_on_Investment by Release_Day_of_Week')
plt.ylabel('Return_on_Investment')
```

Out[174]: Text(0, 0.5, 'Return\_on\_Investment')



```
In [175]: combined_dataset.groupby(['start_year'])['movie'].count().plot(kind='line',  
                                     color="purple",ms=10)  
plt.title('Movie Production between 2010 and 2020')  
plt.ylabel('Number of Movies')
```

Out[175]: Text(0, 0.5, 'Number of Movies')



In [176]:

combined\_dataset

Out[176]:

net_profit_gross	profitable_oversees	genre_ids	popularity	vote_average	vote_count	tconst	original_title	start_year	runtime_minutes	genres
0	0	[18, 9648, 27, 53]	6.099	3.3	102.0	tt3526286	#Horror	2015.0	101.0	Crime,Drama,H
32425665	2013395	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
72082999	36203423	[53, 878, 18]	17.892	6.9	4629.0	tt1179933	10 Cloverfield Lane	2016.0	103.0	Drama,Horror,M
14616	0	[18]	0.955	5.4	7.0	tt3453052	10 Days in a Madhouse	2015.0	111.0	C
38177966	22235984	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	
17800004	24731072	[18, 27, 9648]	15.227	7.0	3458.0	NaN	NaN	NaN	NaN	
141930000	125270000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
44898413	300134946	[28, 12, 80]	21.749	5.6	2452.0	tt1293847	xXx: Return of Xander Cage	2017.0	107.0	Action,Adventure,T
0	895932	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
206678	47094093	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

In [187]:

In [189]:

In [182]:

In [ ]:

```
In [190]:
```

Out[190]:

it_status	domestic_gross	profitable_oversees	genre_ids	popularity	vote_average	vote_count	tconst	original_title	start_year	runtime_minutes	g
Loss	0	0	[18, 9648, 27, 53]	6.099	3.3	102.0	tt3526286	#Horror	2015.0	101.0	C
Profit	32425665	2013395	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	C
Profit	72082999	36203423	[53, 878, 18]	17.892	6.9	4629.0	tt1179933	10 Cloverfield Lane	2016.0	103.0	F
Loss	14616	0	[18]	0.955	5.4	7.0	tt3453052	10 Days in a Madhouse	2015.0	111.0	
Profit	38177966	22235984	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	C
...	...	...	...	...	...	...	...	...	...	...	
Profit	17800004	24731072	[18, 27, 9648]	15.227	7.0	3458.0	NaN	NaN	NaN	NaN	
Profit	141930000	125270000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
Profit	44898413	300134946	[28, 12, 80]	21.749	5.6	2452.0	tt1293847	xXx: Return of Xander Cage	2017.0	107.0	
Loss	0	895932	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
Profit	206678	47094093	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

Merging Datasets

B

Conclusions

## This analysis leads to three recommendations for improving operations of the Austin Animal Center:

- Engage in targeted outreach campaigns for dogs that have been sheltered at AAC for more than 30 days. While most dogs will have been placed after 30 days, this may help reduce the number of dogs that end up having extended stays, potentially requiring many more months of care.
- Reduce current spending until the numbers of intakes and sheltered animals return to normal. Given the reduced activity during this period, AAC should consider ways to temporarily reduce costs by changing space utilization or staffing.
- Hire seasonal staff and rent temporary space for May through December. To accommodate the high volume of intakes and number of sheltered animals in the spring and fall, AAC should leverage seasonal resources, rather than full-year ones. This will allow AAC to cut back on expenditures during the months when there is lower

In [ ]: ▶

In [ ]: ▶