COGS 108 - Movie Popularity and Revenue Analysis

Overview

This project determines if the amount of Google searches of a movie is related to the amount of money it makes. We gathered data from Google and BoxOfficeMojo, containing 206 and 1000 observations respectively. After cleaning our data, we utilized data visualizations, linear prediction, and multivariate regression to conduct our analysis. From this analysis, we discovered a significant positive trend between movie popularity as our predictor variable and movie revenue as our outcome variable.

Names

- Megan Tan
- Nhu Luong
- Ethan Zhou
- David Nguyen
- Jennifer Yang

Research Question

Is there a relationship between the search popularity (in terms of Google searches) a movie receives before its release and the box-office revenue it makes during its opening weekend?

Background & Prior Work

According to Forbes article about how marketing helps box office sales, movie marketers find it difficult to connect specific marketing activities to actual ticket sales because there may be many factors that have influenced a ticket buyer [1]. However, we intend to use the number of Google searches for a movie as a measure of marketing effectiveness. Our beliefs are that a movie's popularity prior to its opening weekend is a positive indication of the success of its gross opening sales.

From our research, we found that television is the most effective method of advertising [2]. Consumers could search up the movie title after viewing the trailer on TV, which could lead to marketing-driven Google searches. While television ranks as one of the more popular vessels of marketing, the Forbes article states that for the typical PG-13 action film during the data's time, shifting 10% of ad budgets from TV to online video could have increased marketing-driven sales by 16% [1]. Searching up movie trailers in Google could also contribute to a spike in trailer views,

leading to increased box office revenue. More recently, major 2017-19 film trailer views on well-known YouTube trailer accounts were counted until the Saturday before the release and were indicative of their high opening weekend box-office revenues. In the data analysis performed on 421 films, including Avengers: Endgame and Avengers: Infinity War, there appeared to be an "upward trend, meaning a positive relationship between YouTube hits and opening weekend ticket sales" [3].

References (include links):

- 1) 8 Insights On How Marketing Drives Movie Box Office Sales https://www.forbes.com/sites/forbesinsights/2015/07/01/8-insights-on-how-marketing-drives-movie-box-office-sales/?sh=698151618ce3
- 2) To Drive Ticket Sales for Action Films, How Should Studios Allocate Their Marketing? https://www.home.neustar/clients/google
- 3) Can YouTube Trailer Views Predict Box Office Openings? https://www.hollywoodreporter.com/news/how-predictive-are-youtube-trailer-views-box-office-openings-1251422

Hypothesis

We predict that there is a positive relationship between Google searches a movie receives and the box-office revenue the movie makes on opening weekend. This means an increase in Google searches corresponds to an increase in revenue a movie makes.

The justification for this hypothesis is that we believe that the more Google searches a movie receives before release, the higher amount of people know about its existence. Therefore, since more people know this movie is coming out, more people will go and see the movie on opening weekend. In a sense, we believe that the number of Google searches a movie gets before it is released is almost equivalent to how well the movie was marketed to consumers. So, logically, the better marketed a movie is, the more money it will make on opening weekend.

Dataset(s)

- Dataset Name: Top Opening Weekends
- Source for the dataset: https://www.boxofficemojo.com/chart/top_opening_weekend/
- Link to the dataset: https://github.com/COGS108/group004_wi21/blob/main/data/opening_weekend.csv
- Number of observations: 1000

This dataset includes all of the opening weekend dates and the box office revenues for the top 1000 movies in the United States from January 1st, 1997 to September 3rd, 2020. The movies are ranked from highest to lowest in terms of their box office revenues on opening weekend. It also contains data about each movie's opening weekend revenue, total gross earning, number of theater it was shown in, release date and distributor.

- Dataset Name: movie_interest.csv

- Source for the dataset: Pulled from Google trends using pytrends
- Link to the dataset: https://github.com/COGS108/group004_wi21/blob/main/data/movie_interest.csv
- Number of observations: 207

This dataset describes the search popularity on Google in the United States for the top 1000 movies based on the top opening weekend revenues in the Top Opening Weekends dataset (sans the duplicate movies), and it lists the movies' relative search popularity for every month from January 1st, 2004 to February 1st, 2021. Search popularity observations are a score from 0 to 100, relative to the highest search popularity, calculated on a monthly basis on Google searches in the United States.

Setup

You may need to restart the kernel after running the following setup

In [197…

```
!pip3 install pytrends
!pip3 install dateparser
!pip3 install BeautifulSoup4 as bs4
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pytrends in ./.local/lib/python3.7/site-packages (4.7.3)
Requirement already satisfied: pandas>=0.25 in /opt/conda/lib/python3.7/site-packages (f
rom pytrends) (1.1.4)
Requirement already satisfied: requests in /opt/conda/lib/python3.7/site-packages (from
pytrends) (2.22.0)
Requirement already satisfied: lxml in ./.local/lib/python3.7/site-packages (from pytren
ds) (4.6.2)
Requirement already satisfied: numpy>=1.15.4 in /opt/conda/lib/python3.7/site-packages
(from pandas>=0.25->pytrends) (1.18.5)
Requirement already satisfied: pytz>=2017.2 in /opt/conda/lib/python3.7/site-packages (f
rom pandas>=0.25->pytrends) (2020.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3.7/site-p
ackages (from pandas>=0.25->pytrends) (2.8.1)
Requirement already satisfied: idna<2.9,>=2.5 in /opt/conda/lib/python3.7/site-packages
(from requests->pytrends) (2.8)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.7/site-packa
ges (from requests->pytrends) (2020.12.5)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /opt/conda/li
b/python3.7/site-packages (from requests->pytrends) (1.25.7)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /opt/conda/lib/python3.7/site-pa
ckages (from requests->pytrends) (3.0.4)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packages (from
python-dateutil>=2.7.3->pandas>=0.25->pytrends) (1.15.0)
WARNING: You are using pip version 20.2.4; however, version 21.0.1 is available.
You should consider upgrading via the '/opt/conda/bin/python3.7 -m pip install --upgrade
pip' command.
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: dateparser in ./.local/lib/python3.7/site-packages (1.0.
0)
Requirement already satisfied: regex!=2019.02.19 in /opt/conda/lib/python3.7/site-packag
es (from dateparser) (2020.10.28)
Requirement already satisfied: pytz in /opt/conda/lib/python3.7/site-packages (from date
parser) (2020.1)
```

```
Requirement already satisfied: python-dateutil in /opt/conda/lib/python3.7/site-packages
(from dateparser) (2.8.1)
Requirement already satisfied: tzlocal in /opt/conda/lib/python3.7/site-packages (from d
ateparser) (2.1)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packages (from
 python-dateutil->dateparser) (1.15.0)
WARNING: You are using pip version 20.2.4; however, version 21.0.1 is available.
You should consider upgrading via the '/opt/conda/bin/python3.7 -m pip install --upgrade
pip' command.
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: BeautifulSoup4 in /opt/conda/lib/python3.7/site-packages
 (4.9.1)
Requirement already satisfied: as in ./.local/lib/python3.7/site-packages (0.1)
Requirement already satisfied: bs4 in ./.local/lib/python3.7/site-packages (0.0.1)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/lib/python3.7/site-packages
 (from BeautifulSoup4) (2.0.1)
WARNING: You are using pip version 20.2.4; however, version 21.0.1 is available.
You should consider upgrading via the '/opt/conda/bin/python3.7 -m pip install --upgrade
pip' command.
```

In [102…

```
import requests
from bs4 import BeautifulSoup
import dateparser
import pandas as pd
from pytrends.request import TrendReq

from datetime import datetime, timedelta
import time

import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import patsy
import statsmodels.api as sm
```

Data Analysis

Data Cleaning:

After importing all the packages necessary to complete our data cleaning, we converted the dollar value given in the opening weekend revenue and the total gross revenue variables from a string to an integer. Next, we initialized a dataframe to hold the opening weekend dataset and renamed/shortened some of the variable names (e.g. “Release” to “Name” of the movie, “Total Gross” to “Total”). We filled the dataframe with data from the Box Office Mojo dataset and stored the table values from the page and stored them into rows. We then extracted relevant information from each row by filling columns with information such as rank, opening, total, open_percent, average, date, and distributor. In a for loop, strings were converted to integers and floats by removing commas and percentages in the ‘opening,’ ‘total,’ and ‘open_percent’ columns, and the dates were converted to datetime formatted dates by using dateparser. We also reset the index to the names of the movies instead of the rank. After scraping this dataset, we saved all the information in the opening_weekend.csv.

We then used the opening weekend dataframe and pytrend to automate and download reports from Google Trends to get information about the interest of each movie over time. Next, we checked if there were any duplicate movie names and dropped the duplicates from the search popularity dataframe (movie_interest.csv). Additionally, we dropped the movies that were released before January 1st, 2004 from the opening weekend dataframe, since Google Trends doesn't cover that timeframe, and the resulting data frame included 766 movies.

When sorting the data into movie_interest.csv, we iterated over all the movies in subsets of [4 different movies, Avengers: Endgame] at a time as build_payload() only permits a maximum of 5 keywords at once. We then used pytrends.build_payload to look for the number of searches for each movie in the subset over the entire data frame. We used temp_df to create a temporary dataframe for those subsets, where each column is a data frame and each row is a list of dates. Each [row, col] value is the relative interest in comparison to the movie, Avengers: Endgame, because it was the top box office movie that we wanted to make the comparisons to. After, we concatenated temp_df to movie_interest_df so that the latter could contain all the subsets and we saved the movie_interest_df to a .csv file.

In [4]:
```
# Converts dollar value given in string to integer
def dollars_to_int(d_string):
    num_str = d_string.replace('$','').replace(',','')
    return int(num_str)
```

In [52]:
```
"""
Dataframe for top 1000 opening weekend box office earning movies.
Saved to file: data/opening_weekend.csv
"""

# Initialize dataframe to hold opening weekend data
opening_weekend_df = pd.DataFrame(columns= ['rank','name', 'opening','total',
                                    'open_percent',
                                    'theaters','average',
                                   'date','dist'])

# Initialize offset to start requested table from (offset 200 returns table from
# boxofficemojo.com starting from rank 201 ending on rank 400)
offset = 0

# Stop when 1000 rows have been scraped
while(offset<1000):

    # URL to scrape from
    URL = 'https://www.boxofficemojo.com/chart/top_opening_weekend/?offset='+str(offset

    # Get table values from page and store in rows
    page = requests.get(URL)
    soup = BeautifulSoup(page.content, 'html.parser')
    rows = soup.find_all('tr')

    # Iterate through all rows excluding first one since it is the table header
    for i in range(1,len(rows)):

        # For each row extract relevant information and standardize
```

```
        row_values = rows[i].find_all('td')

        rank = int(row_values[0].text.replace(',',''))
        name = row_values[1].text

        opening_weekend_str = row_values[2].text
        opening_weekend_int = dollars_to_int(opening_weekend_str)

        total_gross_str = row_values[3].text
        total_gross_int = dollars_to_int(total_gross_str)

        percent_opening_str = row_values[4].text
        percent_opening_float = float(percent_opening_str.replace('%',''))

        theaters_str = row_values[5].text
        theaters_int = int(theaters_str.replace(',',''))

        average_gross_str = row_values[6].text
        average_gross_int = dollars_to_int(average_gross_str)

        release_date = row_values[7].text
        form_release_date = str(dateparser.parse(release_date).date())

        distributor = row_values[8].text.replace('\n','')

        # Create row and insert into opening weekend dataframe
        df_row = [rank,name,opening_weekend_int,total_gross_int,
                  percent_opening_float,theaters_int,average_gross_int,form_release_date
        opening_weekend_df.loc[offset+(i-1)] = df_row

    offset+=200
    print("Finished scraping " + str(offset) + " rows!")


# Verify integrity of opening weekend dataframe
assert(opening_weekend_df.isnull().values.any() == False)
assert(len(opening_weekend_df[opening_weekend_df.opening<0]) == 0)
assert(len(opening_weekend_df[opening_weekend_df.total<0]) == 0)
assert(len(opening_weekend_df[opening_weekend_df.open_percent<0]) == 0)
assert(len(opening_weekend_df[opening_weekend_df.theaters<0]) == 0)
assert(len(opening_weekend_df[opening_weekend_df.average<0]) == 0)


# Remove movies that came out before 01.01.2004 (because no Google search data)
opening_weekend_df['date'] = pd.to_datetime(opening_weekend_df['date'], yearfirst=True)
opening_weekend_df = opening_weekend_df[(opening_weekend_df['date'] >= '2004-01-01 00:0

# Remove duplicates of movies, keeping the higher box office earnings movie
opening_weekend_df.drop_duplicates(subset='name', keep='first', inplace=True)

# Remove movies with unknown/no distributor
opening_weekend_df = opening_weekend_df[(opening_weekend_df['dist'] != '-')]

# Set name column as index column
# opening_weekend_df = opening_weekend_df.set_index('name')

# Save to csv
opening_weekend_df.to_csv('data/opening_weekend.csv')
opening_weekend_df.head(opening_weekend_df.shape[0])
```

```
Finished scraping 200 rows!
Finished scraping 400 rows!
Finished scraping 600 rows!
Finished scraping 800 rows!
Finished scraping 1000 rows!
```

Out[52]:

	rank	name	opening	total	open_percent	theaters	average	date	dist
0	1	Avengers: Endgame	357115007	858373000	41.6	4662	76601	2019-04-26	Walt Disney Studios Motion Pictures
1	2	Avengers: Infinity War	257698183	678815482	38.0	4474	57599	2018-04-27	Walt Disney Studios Motion Pictures
2	3	Star Wars: Episode VII - The Force Awakens	247966675	936662225	26.5	4134	59982	2015-12-18	Walt Disney Studios Motion Pictures
3	4	Star Wars: Episode VIII - The Last Jedi	220009584	620181382	35.5	4232	51987	2017-12-15	Walt Disney Studios Motion Pictures
4	5	Jurassic World	208806270	652270625	32.0	4274	48855	2015-06-12	Universal Pictures
...	...	...	...	...	...	...	...	...	...
992	993	The Manchurian Candidate	20018620	65955630	30.4	2867	6982	2004-07-30	Paramount Pictures
994	995	Pitch Perfect 3	19928525	104897530	19.0	3447	5781	2017-12-22	Universal Pictures
996	997	Ouija	19875995	50856010	39.1	2858	6954	2014-10-24	Universal Pictures
997	998	Kick-Ass	19828687	48071303	41.2	3065	6469	2010-04-16	Lionsgate
998	999	The Unborn	19810585	42670410	46.4	2357	8405	2009-01-09	Universal Pictures

764 rows × 9 columns

In [53]:

```
# Check if there are duplicate movie names
opening_weekend_df = pd.read_csv('data/opening_weekend.csv')
movie_names = opening_weekend_df.loc[:,'name']

try:
    assert len(movie_names) == len(set(movie_names))
    print("There are no duplicate movies.")
```

```
except AssertionError:
    # finding duplicate movies
    movie_counts = dict(map(lambda name: (name, list(movie_names).count(name)), list(mo
    duplicate_movies = [key for (key,value) in movie_counts.items() if value > 1]
    print("There are duplicate movies:", duplicate_movies)
```

```
There are no duplicate movies.
```

In [54]:

```
"""
Dataframe for search popularity (interest) of the top 1000 movies.
Saved to file: data/movie_interest.csv
"""

# get movies list
opening_weekend_df = pd.read_csv('data/opening_weekend.csv')

# remove duplicates
movie_names = list(set(opening_weekend_df.loc[:,'name']))
print("Number of distinct movies after 2004/01/01:", len(movie_names))
```

```
Number of distinct movies after 2004/01/01: 764
```

In [192…

```
# Pulling search keywords from pytrends

# pytrend settings: tz=-480 indicates PST
pytrends = TrendReq(hl='en-US', tz=-480, retries=2, backoff_factor=0.1)
search_dict = {}
i = 0
for name in movie_names:
    i += 1
    if (i%100 == 0):
        print("Finished " + str(i) + " movies")
    try:
        suggestions = pytrends.suggestions(name)
        for suggestion in suggestions:
            if 'type' in suggestion and 'film' in suggestion['type'].lower() and 'mid'
                search_dict[name] = suggestion['mid']
                break
    except:
        # query returns error on Farenheit 9/11, so it is excluded
        print(name)
print(len(search_dict))
search_dict
```

```
Fahrenheit 9/11
Finished 100 movies
Finished 200 movies
Finished 300 movies
Finished 400 movies
Finished 500 movies
Finished 600 movies
Finished 700 movies
730
```

Out[192…

```
{'Vantage Point': '/m/02rrh1w',
 'Cars': '/m/03q0r1',
 'Big Hero 6': '/m/0v3h5y7',
 'xXx: Return of Xander Cage': '/m/083xgxs',
 "Miss Peregrine's Home for Peculiar Children": '/g/12lr2z_p0',
 'Friday the 13th': '/m/0_5c7kr',
```

```
'The Twilight Saga: Breaking Dawn - Part 1': '/m/075wx89',
'The Other Woman': '/m/05f7jsw',
'Skyfall': '/m/02vxq9m',
'The Break-Up': '/m/078958',
'White Noise': '/m/0g514rw',
'This Is the End': '/m/0hz4j2j',
'Law Abiding Citizen': '/m/05h4_qt',
'Avatar': '/m/0bth54',
'Journey 2: The Mysterious Island': '/m/0ds2_k2',
'Sausage Party': '/m/0zwqnxs',
"Lee Daniels' The Butler": '/m/0lkvn87',
'X-Men Origins: Wolverine EXTENDED': '/m/0fpqrp8',
'Pain & Gain': '/m/0gy15f9',
'Looper': '/m/0by1wkq',
"The Hitchhiker's Guide to the Galaxy": '/m/03wjm2',
'The Day After Tomorrow': '/m/024l2y',
'Star Trek Beyond': '/m/010kk6fr',
'Tammy': '/m/0w1f1l8',
'The SpongeBob SquarePants Movie': '/m/03vfr_',
'The Conjuring': '/m/0j2drrs',
'The Twilight Saga: New Moon': '/m/05pdh86',
'Despicable Me': '/g/11fzfc279x',
'Thor': '/m/0j9knfv',
'The Hunger Games: Mockingjay - Part 1': '/m/0ngvsvk',
'Sully': '/g/11bw6qksy0',
'The Strangers': '/g/11ckxwqfvy',
'Fifty Shades of Grey': '/m/0jx8jz8',
'The Dark Knight': '/m/0btpm6',
'Transformers': '/m/0czdmfg',
'World War Z': '/m/0gyvj60',
'The Nun': '/g/11dfhx0ryn',
'Madea Goes to Jail': '/m/03wgvhl',
'Wanted': '/m/0287477',
'The Muppets': '/m/0y50xvq',
'Terminator: Dark Fate': '/g/11fy2lq7ml',
'Poseidon': '/m/07y21m',
'Insidious: Chapter 2': '/m/0vpprvm',
'Evil Dead': '/m/03d81nn',
'Gravity': '/g/11cnxz1kq3',
'John Wick: Chapter 3 - Parabellum': '/g/11f148l14l',
'About Last Night': '/m/0t_dn7n',
'Think Like A Man Too': '/m/0w1df97',
'The SpongeBob Movie: Sponge Out of Water': '/m/0jt9rxr',
'Murder on the Orient Express': '/g/11c2kjljm2',
'Jack and Jill': '/m/0ch3qr1',
'10,000 BC': '/m/0bqvqq',
'Tomorrowland': '/m/0qfnvgz',
'A Wrinkle in Time': '/g/11c5h3v9j3',
'The Nutcracker and the Four Realms': '/g/11c71c0t1d',
'Super 8': '/m/04wzkz',
'The Lego Movie 2: The Second Part': '/g/11c311_1w1',
'The Secret Life of Pets 2': '/g/11f65957s5',
'TRON: Legacy': '/m/05zy2cy',
"Michael Jackson's This Is It": '/m/076xv29',
'Ice Age: Continental Drift': '/m/0fpgp26',
'G-Force': '/m/03bx2lk',
'Minions': '/m/0tl3fpv',
'The Vow': '/m/0dscrwf',
'SAW: The Final Chapter': '/m/09p5mwg',
'Solo: A Star Wars Story': '/g/11c342w31k',
'Superman Returns': '/m/044g_k',
'Bee Movie': '/m/087m22',
'Annabelle': '/m/0_lmjt2',
'The 40-Year-Old Virgin': '/m/06fpsx',
'The Polar Express': '/m/09w6br',
```

```
'Edge of Tomorrow': '/m/0hrdg1w',
'Grown Ups': '/m/0640m69',
'2012': '/m/047vnkj',
'Captain America: The Winter Soldier': '/m/0jzt8tx',
'Race to Witch Mountain': '/m/04f76fm',
'Garfield': '/m/01yzvd',
'Couples Retreat': '/m/04zwwhf',
'The Da Vinci Code': '/m/065dc4',
'Percy Jackson & the Olympians: The Lightning Thief': '/m/0642xf3',
'Aquaman': '/m/0126dxpv',
'Baby Driver': '/g/11c4b67662',
'Constantine': '/m/04hk0w',
'Robots': '/m/03mh94',
'Ride Along 2': '/m/0114jq6n',
'Captain America: The First Avenger': '/m/0cc846d',
'The Grudge 2': '/g/11hcz1qql8',
'A Good Day to Die Hard': '/m/0hhgg_1',
'Hellboy': '/g/11dfl0wyc3',
'Hotel Transylvania 3: Summer Vacation': '/g/11c6dl0vy0',
'10 Cloverfield Lane': '/m/0hxxbp3',
'Now You See Me 2': '/m/011djb5_',
'Bumblebee': '/g/11dymbqf4b',
'Thor: Ragnarok': '/m/0126b7q0',
'Thor: The Dark World': '/m/0j9knfv',
'Beverly Hills Chihuahua': '/m/0l8h5rw',
'The Addams Family': '/g/11f661nr0y',
'The Upside': '/g/11c327xhv8',
'The Green Hornet': '/m/07k8rt4',
'Nacho Libre': '/m/07d3fs',
'Furious 7': '/m/0vsjsxr',
'Megamind': '/m/05229c_',
'Angels & Demons': '/m/0f4_2k',
"Fool's Gold": '/m/03c_lhp',
'War of the Worlds': '/m/02mmwk',
'Sin City': '/m/04tqtl',
'Sing': '/m/02mz_6',
'Night School': '/g/11gbk4ryc2',
'Cars 2': '/m/0407yj_',
'The Wedding Ringer': '/m/0y4s003',
'The Divergent Series: Allegiant': '/g/11b7v4vnqj',
'Indiana Jones and the Kingdom of the Crystal Skull': '/m/04mcw4',
'Us': '/m/02771cl',
'Jumanji: Welcome to the Jungle': '/g/11c1rg68rq',
'Norbit': '/m/0kv2hv',
'District 9': '/m/0581vn8',
'Bad Grandpa': '/m/0wfl62_',
"Disney's A Christmas Carol": '/m/02vqhv0',
'Gnomeo & Juliet': '/m/087ksg',
'Beauty and the Beast': '/g/11b7d_21j0',
'Journey to the Center of the Earth': '/m/0ds2_k2',
'Four Christmases': '/m/03gttvn',
'The Ugly Truth': '/m/047rkcm',
'Doctor Strange': '/g/11fmhx17vl',
'Bad Boys for Life': '/g/11bx44mqwp',
'Fast & Furious Presents: Hobbs & Shaw': '/g/11fctqqzrc',
'I Am Legend': '/m/0gfsq9',
'Dawn of the Dead': '/m/02my7z',
'The Good Dinosaur': '/m/0jwmwgc',
'Van Helsing': '/m/02nx2k',
'Downton Abbey': '/g/11fj7hqdmf',
'Mr. & Mrs. Smith': '/m/05sns6',
'Pirates of the Caribbean: Dead Men Tell No Tales': '/m/0h_dv23',
'Just Go with It': '/m/0bshwmp',
'Spider-Man: Into the Spider-Verse': '/g/11j0wlxbj_',
'The Princess Diaries 2: Royal Engagement': '/m/036p6v',
```

```
'Son of God': '/m/0x0v5mf',
'Ice Age: Collision Course': '/g/11b7ltkhwf',
'Step Brothers': '/m/026mfbr',
'Diary of a Wimpy Kid: Rodrick Rules': '/m/0g5r4vt',
'The Bourne Supremacy': '/m/03k8th',
'Alien: Covenant': '/g/1q6lfvj8s',
'G.I. Joe: The Rise of Cobra': '/m/03qcfvw',
'Unbroken': '/m/0wjk75w',
'Teenage Mutant Ninja Turtles': '/m/073v9d',
'Annabelle: Creation': '/g/11b_02llj_',
"TYLER PERRY'S A MADEA FAMILY FUNERAL": '/g/11ffv13crs',
'Dear John': '/m/05b08zq',
'Marley & Me': '/m/0q3_01k',
'High School Musical 3': '/m/0286_hv',
'G.I. Joe: Retaliation': '/m/0gwlfnb',
"The Hitman's Bodyguard": '/g/11cmrws3vr',
'Salt': '/m/05pbl56',
'Kung Fu Panda 2': '/m/06ztvyx',
'Despicable Me 3': '/g/11fzfc279x',
'Dunkirk': '/g/11bxfwfy23',
'Real Steel': '/m/0c3zjn7',
'Tropic Thunder': '/m/02ryz24',
'Resident Evil: Afterlife': '/m/080dfr7',
'Maze Runner: The Death Cure': '/g/11cm00twww',
'Cars 3': '/m/010gz3f0',
'Due Date': '/m/06_w90f',
'Friday Night Lights': '/m/042zrm',
'Inside Man': '/m/09nl36',
'Madagascar: Escape 2 Africa': '/m/0b3n61',
'X-Men: Apocalypse': '/m/0_4654w',
'Be Cool': '/m/056xkh',
"Ender's Game": '/m/05p1_m',
'Jackass Number Two': '/m/0df2zx',
'Ratatouille': '/m/03x7hd',
'The Curse of la Llorona': '/g/11f3w6p1zx',
'Onward': '/g/11fj328t_2',
'Harry Potter and the Deathly Hallows: Part 2': '/m/0gvsynb',
'Maleficent': '/m/0k0tq97',
'San Andreas': '/m/010fl0v_',
'The Invisible Man': '/g/11hz8nk82d',
'Ant-Man': '/m/0bbcrq',
'Godzilla': '/m/0ryv9d1',
'The Purge: Election Year': '/g/11b6wqp7vs',
'Neighbors': '/m/0j_3rdp',
'Anchorman 2: The Legend Continues': '/m/0jws4d1',
'Troy': '/m/01vw8k',
'London Has Fallen': '/m/010vzxsl',
'Iron Man 3': '/m/0bc1yhb',
'The Hunger Games: Catching Fire': '/m/0gkz15s',
'Insidious: The Last Key': '/g/11cs1kw12y',
'Goosebumps': '/m/052w963',
'Maleficent: Mistress of Evil': '/g/11f647wxry',
'Sherlock Holmes': '/m/04n52p6',
'Miami Vice': '/m/07p12s',
'Magic Mike': '/m/0h63gl9',
'All Eyez on Me': '/g/11bxfpxyng',
'How to Train Your Dragon': '/m/04f52jw',
'Total Recall': '/m/011wtv',
'The Peanuts Movie': '/m/0100n5b8',
"Pirates of the Caribbean: At World's End": '/m/05nlx4',
'Pacific Rim': '/g/11c2kjsd26',
'Bewitched': '/m/0102n_tv',
'Man on Fire': '/m/02lk42',
'The Amazing Spider-Man 2': '/m/0qkv9wb',
'John Wick: Chapter 2': '/g/11bydmrh02',
```

```
'The Incredibles': '/m/02qm_f',
'Captain Marvel': '/m/0126b88c',
'Kick-Ass': '/m/05f4_n0',
'Hellboy II: The Golden Army': '/m/0dcz8_',
'Blades of Glory': '/m/0gldyz',
'Ted 2': '/m/0114hb3m',
'Dolittle': '/g/11ggvyyn8l',
'Star Trek Into Darkness': '/m/0hhqv27',
'Battle Los Angeles': '/m/06zt03v',
'What Happens in Vegas': '/m/03c3vl6',
'Meet the Browns': '/m/03cnlkv',
'Elysium': '/m/0gwm_wy',
'Fast & Furious 6': '/m/0gtxbqr',
'Terminator Salvation': '/m/043tvp3',
'Now You See Me': '/m/011djb5_',
'The Haunting in Connecticut': '/m/043s2w5',
'The Mummy: Tomb of The Dragon Emperor': '/m/028cg00',
'50 First Dates': '/m/02f6g5',
'Inglourious Basterds': '/m/02yvct',
'Where the Wild Things Are': '/m/09txzv',
'Dracula Untold': '/m/0wkcg_5',
'Pet Sematary': '/g/11fctqtwd6',
'The Game Plan': '/m/0kvfv3',
'Barbershop: The Next Cut': '/m/01370796',
'The Best Man Holiday': '/m/0mzxvjs',
'The Maze Runner': '/m/012dvzxm',
'The Predator': '/g/11bxw2b6y5',
'Toy Story 3': '/m/04hwbq',
'The Fast and the Furious: Tokyo Drift': '/m/08c6k9',
'The Wolfman': '/m/0kv238',
'Public Enemies': '/m/0g3zrd',
'Rush Hour 3': '/m/06x43v',
'Ghost Rider: Spirit of Vengeance': '/m/0dgq80b',
'Aladdin': '/m/06_w18q',
'Quantum of Solace': '/m/08gsvw',
'It Chapter Two': '/g/11hblcl111',
'Snow White and the Huntsman': '/m/0g5qs2k',
'The Legend of Tarzan': '/m/0w17g_l',
'Batman v Superman: Dawn of Justice': '/m/0wrshm2',
'The Divergent Series: Insurgent': '/m/0_qvrhs',
'Iron Man': '/m/0dzlbx',
'Gone Girl': '/m/0w91n0z',
'Christmas With the Kranks': '/m/04l5bx',
'Paranormal Activity 3': '/m/0h1f30d',
'The Unborn': '/m/03wgd47',
'The Blind Side': '/m/05zy3sc',
'Grown Ups 2': '/m/0mzt5cs',
'Ladder 49': '/m/03__xl',
'The Fate of the Furious': '/g/11bc6_2t_7',
'Knowing': '/m/047v2p4',
'Brave': '/m/05mvcz5',
'Ralph Breaks the Internet': '/g/11c0xht716',
'Angel Has Fallen': '/g/11f3w6s6y5',
'Alvin and the Chipmunks: The Squeakquel': '/m/0640y35',
'Die Hard 4.0': '/m/05t54s',
'The Bourne Ultimatum': '/m/061681',
'I, Robot': '/m/02qkwl',
'2 Guns': '/m/0nb2hgq',
'Superbad': '/m/0fz3b1',
'The Taking of Pelham 123': '/m/03wbqc4',
'V for Vendetta': '/m/0645k5',
'Ford v Ferrari': '/g/11f6y3pfry',
'Chicken Little': '/m/04g73n',
'Blockers': '/g/11dyqy9c6v',
'Fast & Furious': '/m/0czbbsz',
```

```
'Pirates of the Caribbean: On Stranger Tides': '/m/09v8clw',
"Daddy's Home 2": '/g/11c74fmp0k',
'Straight Outta Compton': '/m/011djj66',
'Prometheus': '/m/0gd0c7x',
'A Star Is Born': '/m/0g9t50b',
'Diary of a Wimpy Kid': '/m/080jm_y',
'Monster House': '/m/07y9w5',
'American Gangster': '/m/0cz_ym',
'Notorious': '/m/01ltq_',
'Wedding Crashers': '/m/04yc76',
'National Treasure: Book of Secrets': '/m/09wnnb',
'Kill Bill: Volume 2': '/m/0gyv0b4',
'Pitch Perfect 3': '/g/11b7xszwdx',
'Paranormal Activity 2': '/m/0bv6qnt',
'Walk the Line': '/m/05jzt3',
'Mission: Impossible III': '/m/06fqlk',
'Hotel Transylvania': '/m/0gj96ln',
"Madea's Big Happy Family": '/m/0djb4gn',
'Chronicle': '/m/07vn_9',
'Zombieland: Double Tap': '/m/0cj0v2b',
'Spy': '/m/0bkbm',
'Godzilla: King of the Monsters': '/g/11c6gz2d_5',
'X-Men: Dark Phoenix': '/g/11d_8843w5',
'WALL-E': '/m/027s39y',
'Underworld Evolution': '/m/0548xg',
'Trainwreck': '/m/010phc21',
'Smallfoot': '/g/11g88hkzc1',
'Why Did I Get Married?': '/m/02731xp',
'Star Wars: Episode III - Revenge of the Sith': '/m/0fdv3',
"Ocean's Thirteen": '/m/09xbpt',
'Creed II': '/m/012nmptp',
'Failure to Launch': '/m/0b9qwr',
'The Perfect Guy': '/m/0bggk69',
'The A-Team': '/m/07_k0c0',
'Disturbia': '/m/0dmqyj',
'The Purge': '/m/011q3ysq',
'Barbershop 2: Back in Business': '/m/02ntj8',
"Tyler Perry's Temptation: Confessions of A Marriage Counselor": '/m/0h668xg',
'The Hobbit: The Battle of the Five Armies': '/m/0ljff3c',
'The Monuments Men': '/m/0ndx3rt',
'Deadpool': '/m/062zn4b',
'Get Out': '/g/11c1c4yplq',
'Jason Bourne': '/m/04cxzzb',
'Rise of the Planet of Apes': '/m/0n05bkp',
'Warcraft': '/m/0wkd2g3',
'Lights Out': '/g/11btymdphw',
'A Nightmare On Elm Street': '/m/08ll52',
'27 Dresses': '/m/03c5cgj',
'The Grudge': '/g/11hcz1qql8',
'Teenage Mutant Ninja Turtles: Out of the Shadows': '/g/11b67yc985',
'The Stepford Wives': '/m/034qbx',
'Open Season': '/m/086tz1',
'Tomb Raider': '/m/0d6_s',
'Bruno': '/m/026vmql',
'The Expendables': '/m/053rxgm',
'Frozen II': '/g/11btxgryx0',
'Jackass 3D': '/m/09rx7tx',
'The Pacifier': '/m/057jdv',
'Captain Phillips': '/m/0hyq1j4',
'The Great Gatsby': '/m/0gtt5fb',
'Harry Potter and the Half-Blood Prince': '/m/03hxsv',
'Rogue One: A Star Wars Story': '/g/11b7ck8r7s',
'Taken 2': '/m/0hhggmy',
'The Accountant': '/m/012nlb0j',
'The Longest Yard': '/m/04flr5',
```

```
'Kong: Skull Island': '/g/11bwcryqqd',
'The Manchurian Candidate': '/m/03shpq',
'Fury': '/m/0t_5p_m',
'Robin Hood': '/m/01s0l0',
'Identity Thief': '/m/0lqhxhk',
'Avengers: Endgame': '/m/0126b8kv',
'300': '/m/07f_t4',
'The Equalizer 2': '/g/11f11pply3',
'Silent Hill': '/m/07f_7h',
'The Hunger Games': '/m/0ngvsvk',
'The Heat': '/m/0k95mpr',
'Predators': '/m/064k1wb',
'The Boss Baby': '/m/01281c_d',
'National Treasure': '/m/033srr',
'Non-Stop': '/m/0ng5055',
'The Hobbit: The Desolation of Smaug': '/m/052nd_v',
'The Departed': '/m/04vr_f',
'The Croods': '/m/010pch65',
'The Lucky One': '/m/0hgky8t',
'Warm Bodies': '/m/0gxv5h3',
'Transformers: Revenge of the Fallen': '/m/047csmy',
"Madagascar 3: Europe's Most Wanted": '/m/05c26ss',
'The Chronicles of Narnia: The Voyage of the Dawn Treader': '/m/0642ykh',
'Boo! A Madea Halloween': '/g/11c3wtypym',
'Prisoners': '/g/11hdn81x65',
'Annabelle Comes Home': '/g/11ffg1qtb8',
'Knocked Up': '/m/0bvn25',
'Knives Out': '/g/11fk8dh5w5',
'Pitch Perfect 2': '/m/0y4nmtc',
'21': '/m/02qczf7',
'The Call of the Wild': '/g/11ffmnbrzr',
'Jack the Giant Slayer': '/m/0glqw0c',
'Abominable': '/g/11f66b0pdz',
'Man of Steel': '/m/0gjc4d3',
'Men in Black: International': '/g/11f730y46p',
'Blade Runner 2049': '/g/11b6jk94gw',
'The Campaign': '/m/0hgnvc3',
'Jurassic World': '/m/051179',
'Along Came Polly': '/m/02nt3d',
'Terminator Genisys': '/m/0v_nfcl',
'Interstellar': '/m/0fkf28',
'Sherlock Holmes: A Game of Shadows': '/m/0dlngsd',
'Happy Feet': '/m/05650n',
'The Chronicles of Riddick': '/m/031gmx',
"We're the Millers": '/m/0lgq7_r',
'No Good Deed': '/g/11f01zprwr',
'John Carter': '/m/03whyr',
'Shrek Forever After': '/m/02qydsh',
'Kung Fu Panda': '/m/09146g',
'The Wolverine': '/g/11cn3h3kmj',
'17 Again': '/m/03d79mp',
'Alita: Battle Angel': '/m/08tyfg',
'The Jungle Book': '/m/0110nff9',
'The Hobbit: An Unexpected Journey': '/m/0ndwt2w',
'Little Fockers': '/m/09g8vhw',
'Pacific Rim: Uprising': '/g/11c2kjsd26',
'Ghostbusters': '/g/11fjn2v02_',
'Rampage': '/g/11df0xjphv',
'Kingsman: The Secret Service': '/m/0y4n5ll',
'Ouija': '/g/11bw50r5pp',
'Once Upon a Time... In Hollywood': '/g/11f4rdmxqn',
'The Mummy': '/m/01ln5z',
'Cloudy with a Chance of Meatballs': '/m/05c1x2g',
'Eragon': '/m/05c9zr',
'American Reunion': '/m/0gtsx8c',
```

```
 'The Amazing Spider-Man': '/m/0fqt1ns',
 'The Twilight Saga: Breaking Dawn - Part 2': '/m/0djz0rc',
 'Get Hard': '/m/0100c6vp',
 'The Devil Inside': '/m/0h96hd7',
 'Gemini Man': '/g/11ggbbgwz8',
 'Beowulf': '/m/09210y',
 'Spider-Man: Homecoming': '/m/0wkgbxq',
 'The Woman in Black': '/m/0dgnnp1',
 'Borat: Cultural Learnings of America for Make Benefit Glorious Nation of Kazakhstan':
'/m/0dt8xq',
 'Bride Wars': '/m/026y3vk',
 'Heaven Is for Real': '/m/0wjx8tt',
 'Star Wars: Episode VIII - The Last Jedi': '/m/0t_7k4v',
 'Pixels': '/m/010rdrd1',
 'X-Men: First Class': '/m/0cd2vh9',
 'Taken 3': '/m/0wry9ll',
 'Tower Heist': '/m/0dlns08',
 "Big Momma's House 2": '/m/0811_b',
 'Date Night': '/m/0g7pm1',
 'Bedtime Stories': '/m/02qdrjx',
 'Tangled': '/m/02xbyr',
 'The Hangover Part III': '/m/0n3xxpd',
 'Joker': '/m/0v3b_f8',
 'War for the Planet of the Apes': '/m/0113z663',
 'Independence Day: Resurgence': '/m/0134v4hd',
 'Bad Teacher': '/m/0cp0ph6',
 'Rio 2': '/m/0j9m0k2',
 'Funny People': '/m/047svrl',
 'A Series of Unfortunate Events': '/m/04k9y6',
 'Jack Reacher: Never Go Back': '/g/11b6s_zq9p',
 'The Lego Batman Movie': '/g/11c264g0v0',
 'Ghost Rider': '/m/06yykb',
 'Captain Underpants: The First Epic Movie': '/m/0115c213',
 'Good Boys': '/g/11f7r5xwtq',
 "Pete's Dragon": '/m/012n9rw5',
 'Exodus: Gods and Kings': '/m/0xpqbly',
 'Rise of the Guardians': '/m/06wbm8q',
 'Black Panther': '/m/0126b7rc',
 'Les Misérables': '/m/0h51_mt',
 'The Chronicles of Narnia: Prince Caspian': '/m/063fh9',
 'Hustlers': '/g/11h6dctfb9',
 'Deadpool 2': '/g/11cmqkzqvq',
 'The Angry Birds Movie': '/m/0110gltf',
 'The Forgotten': '/m/04153z',
 'Meet the Robinsons': '/m/06fcqj',
 'Insidious: Chapter 3': '/m/011ccb71',
 'Oblivion': '/m/0_fpx3l',
 'Scary Stories to Tell in the Dark': '/g/11fkmx0x4s',
 'A Quiet Place': '/g/11fktrhwr_',
 "Ocean's Twelve": '/m/0418wg',
 'Sex and the City: The Movie': '/m/03bzyn4',
 'Mr. Peabody & Sherman': '/m/0gwq0hb',
 'Hide and Seek': '/m/04fyc2',
 'Boo 2! A Madea Halloween': '/g/11g8cytwhd',
 'Deepwater Horizon': '/g/11b6b4yz64',
 'The Emoji Movie': '/g/11cn3jwh3n',
 'Planes': '/m/0hgm9yp',
 'Eight Below': '/m/07yp01',
 'Hannah Montana: The Movie': '/m/02x3lt7',
 'The Karate Kid': '/m/0mmbb',
 'Christopher Robin': '/g/11f0_blgtm',
 '300: Rise of an Empire': '/m/0jt3mlt',
 'Happy Feet Two': '/m/09rvvpm',
 'I Can Do Bad All by Myself': '/m/05b_8_1',
 'Hitch': '/m/0j_c',
```

```
'The Book of Eli': '/m/04ydr95',
'Saw IV': '/m/02pcq92',
'Fifty Shades Darker': '/g/11b76gb5h5',
'Monsters Vs. Aliens': '/m/02qqs_k',
'Dark Shadows': '/m/02rnh3f',
'Flightplan': '/m/06dq4r',
'Hop': '/m/0ch3jlt',
'Django Unchained': '/m/0gwjw0c',
'Immortals': '/m/076tw54',
'Penguins of Madagascar': '/m/0zwpwtt',
'The Secret Life of Pets': '/m/0115c2x3',
'Puss in Boots': '/m/0cmf0m0',
'Takers': '/m/04f285v',
'Black Mass': '/m/010pd7hh',
'Ice Age: The Meltdown': '/m/0729rh',
'Spider-Man: Far from Home': '/g/11d_7x5tjx',
'Saw III': '/m/09rfh9',
'Wonder Woman': '/g/11gdrdv3l3',
'Star Wars: Episode IX - The Rise of Skywalker': '/m/0t_7k53',
'Green Lantern': '/m/0bh8yn3',
'Power Rangers': '/m/0344xk',
'Scooby-Doo 2: Monsters Unleashed': '/m/036fxp',
'The Purge: Anarchy': '/m/0w_scxd',
'Pineapple Express': '/m/02825nf',
'Casino Royale': '/m/03r0g9',
'King Kong': '/m/02dr9j',
'Spider-Man 3': '/m/0340hj',
'Alice in Wonderland': '/m/04jpg2p',
'Justice League': '/m/010fq514',
'The Ring Two': '/m/04xx35',
'The House with a Clock in Its Walls': '/g/11f_p2m72w',
'Rocketman': '/m/0dkhyx',
'Resident Evil: Extinction': '/m/06_sc3',
'Fantastic Beasts and Where to Find Them': '/g/11b5v9gx4p',
'Monsters University': '/m/0gwndqr',
'The Town': '/m/07kh6f3',
'Birds of Prey': '/g/11fd6dh23r',
'Into the Woods': '/m/0w2sx8x',
'Taken': '/m/0hhggmy',
'Contagion': '/m/0dgnwwr',
'Final Destination 3': '/m/0680y4',
'Batman Begins': '/m/0btpm6',
'The Devil Wears Prada': '/m/0cbv4g',
'Stomp the Yard': '/m/0270t4y',
'The Chronicles of Narnia: The Lion, the Witch and the Wardrobe': '/m/0639bg',
'Starsky & Hutch': '/m/02ntb8',
'Kung Fu Panda 3': '/m/0c03gcc',
'Up': '/m/02rn00y',
'The Help': '/m/0ds3t5x',
'The Avengers': '/m/062zm5h',
'Dumb And Dumber To': '/m/0372j5',
'Meet the Fockers': '/m/03tps5',
'Underworld Awakening': '/m/084h00h',
'The Hangover': '/m/05p1tzf',
'Trolls': '/g/11flzl65lb',
'Cloudy with a Chance of Meatballs 2': '/m/0nbn1x_',
'Paul Blart: Mall Cop': '/m/03why2r',
'Wonder': '/g/11gdrdv3l3',
'The Curious Case of Benjamin Button': '/m/026p4q7',
'Battleship': '/m/0bh8tgs',
'Shutter Island': '/m/03cp4cn',
'Transformers: Age of Extinction': '/m/0mzgk6d',
'Eagle Eye': '/m/03ct7jd',
'Monster-in-Law': '/m/063drh',
'How to Train Your Dragon: The Hidden World': '/m/0m_cnk_',
```

```
'Ride Along': '/m/0swlzl6',
'13 Going on 30': '/m/02vzpb',
'Transformers: Dark of the Moon': '/m/0872p_c',
'Hotel Transylvania 2': '/m/0118fw_v',
'Bohemian Rhapsody': '/g/11c5m5019b',
"It's Complicated": '/m/076tq0z',
'Avengers: Infinity War': '/m/0126b8kn',
'Inception': '/m/0661ql3',
'Harry Potter and the Prisoner of Azkaban': '/m/03177r',
'Harry Potter and the Order of the Phoenix': '/m/031hcx',
'The Visit': '/m/011q3zfx',
'Dinner for Schmucks': '/m/03h4fq7',
'Charlie and the Chocolate Factory': '/m/04pk1f',
'Mamma Mia! Here We Go Again': '/g/11fy55b70g',
'Contraband': '/m/0glqh5_',
'SAW 2': '/m/05jwph',
'The Other Guys': '/m/087vnr5',
'Moana': '/g/11b6pr8_4f',
'Skyscraper': '/g/11g9nrfqkp',
'White House Down': '/m/0n4blqr',
'Spider-Man 2': '/m/02wgk1',
'X-Men: Days of Future Past': '/m/0r3r5jz',
'Hancock': '/m/02vrgnr',
'The Hunger Games: Mockingjay - Part 2': '/m/0ngvtb_',
'Mary Poppins Returns': '/g/11cs97d368',
'Transformers: The Last Knight': '/g/11cnd8ky2x',
'The Girl on the Train': '/g/11bwn3z58n',
'Star Wars: Episode VII - The Force Awakens': '/m/0nb2r_p',
'Bad Moms': '/g/11g7_mt0p3',
'Harry Potter And The Deathly Hallows - Part 1': '/m/02pth35',
"Valentine's Day": '/m/06_wqk4',
'Maze Runner: The Scorch Trials': '/m/011r9j66',
'Bridge to Terabithia': '/m/026gt8b',
'I Now Pronounce You Chuck & Larry': '/m/0f2sx4',
'The Pursuit of Happyness': '/m/06rhz7',
'Valkyrie': '/m/02qhlwd',
'When a Stranger Calls': '/m/08k8v0',
'Eat Pray Love': '/m/07kdkfj',
'X-Men: The Last Stand': '/m/06gb1w',
'Hairspray': '/m/0bt3j9',
'Jurassic World: Fallen Kingdom': '/g/11c6t8rk5z',
'Oz the Great and Powerful': '/m/0gysshc',
'The Golden Compass': '/m/04w7rn',
'Sex and the City 2': '/m/05h43ls',
'A Madea Family Funeral': '/g/11ffv13crs',
'Inside Out': '/m/0wpwlgl',
'Underworld: Rise of the Lycans': '/m/03h44mv',
'Venom': '/m/0856pm',
'How to Train Your Dragon 2': '/m/0gwp_k3',
'The Bourne Legacy': '/m/03qnvdl',
'The Smurfs': '/m/0n_bn4d',
'Mad Max: Fury Road': '/m/02d3wv',
'Why Did I Get Married Too': '/m/063zc8f',
'Paul Blart: Mall Cop 2': '/m/010fch95',
'Step Up': '/m/0b73wst',
'Shazam!': '/m/0126b88c',
'Divergent': '/m/0v93qv0',
'Central Intelligence': '/g/11b726t1bj',
'Zookeeper': '/m/07kb7vh',
'The Village': '/m/02q87z6',
'The Martian': '/m/01295z79',
'You, Me and Dupree': '/m/08lwpy',
'The Hangover Part II': '/m/0dln8jk',
'Cinderella': '/g/11cst1fhg5',
'Pokémon Detective Pikachu': '/g/11f3ws3v46',
```

```
'Night at the Museum: Battle of the Smithsonian': '/m/04gv3db',
'The Forbidden Kingdom': '/m/0gj6pd',
'True Grit': '/m/0b73_1d',
'Guardians of the Galaxy': '/m/0ryw0v0',
'The Interpreter': '/g/11gfm7t7gf',
'The Meg': '/g/11c1wmxycd',
'The Dark Knight Rises': '/m/0bpm4yw',
'Wild Hogs': '/m/0f1vgm',
'Fantastic Beasts: The Crimes of Grindelwald': '/g/11b5v7k53f',
'Arrival': '/m/0bswf7',
'The Amityville Horror': '/m/0w_tdg4',
'Wreck-It Ralph': '/m/0cc97st',
'Diary Of A Mad Black Woman': '/m/0570h6',
'The Passion of The Christ': '/m/01br2w',
'Talladega Nights: The Ballad of Ricky Bobby': '/m/08952r',
'Saw V': '/m/0fq7dv_',
'Zootopia': '/m/0128rxy6',
'Noah': '/m/047tsx3',
'Resident Evil: Apocalypse': '/m/03cwwl',
'Captain America: Civil War': '/m/0125zrjx',
'The Lorax': '/m/087wc7n',
'The Equalizer': '/m/0v16d9t',
'Little Man': '/m/0bbqyd',
'The Bounty Hunter': '/m/06w9zfg',
'Olympus Has Fallen': '/m/0n43ym4',
'The Lone Ranger': '/m/0dsf7lf',
"He's Just Not That Into You": '/m/03bzjpm',
'Poltergeist': '/m/0_t02',
'Mission: Impossible - Fallout': '/g/11c75wzc6t',
'Tenet': '/g/11h193b_c5',
'Anchorman: The Legend of Ron Burgundy': '/m/034qzw',
'The Simpsons Movie': '/m/050f0s',
'Deja Vu': '/m/090w3m',
'The Social Network': '/m/07s846j',
'The Last Exorcism': '/m/0bh9zt4',
'Suicide Squad': '/m/012dwdjr',
'My Bloody Valentine': '/m/03qmt28',
'The Happening': '/m/02pxmgz',
'Storks': '/g/11bwnd3yzf',
'Watchmen': '/m/0czyxs',
'Texas Chainsaw': '/m/0h1ckhz',
'Get Smart': '/m/026wlxw',
'Men in Black 3': '/m/0661m4p',
'Horrible Bosses': '/m/0crd8q6',
"You Don't Mess with the Zohan": '/m/02ph9tm',
'Incredibles 2': '/m/0_yxzlv',
'Cowboys & Aliens': '/m/0cc5mcj',
'Clash of the Titans': '/m/05szq8z',
"Dr. Seuss' Horton Hears a Who!": '/m/0fk25m',
'RoboCop': '/m/0299hs',
'Happy Death Day': '/g/11dftc92c_',
'Justin Bieber: Never Say Never': '/m/0g9tcbg',
'Enchanted': '/m/0d4htf',
'Despicable Me 2': '/m/06zkfsy',
'Over the Hedge': '/m/080_x_',
'The Twilight Saga: Eclipse': '/m/075wx7_',
'Rango': '/m/06w99h3',
'Mama': '/m/0hynmlk',
'Dumbo': '/g/11fxdy6r7g',
'Obsessed': '/m/07s43gs',
'Jumanji: The Next Level': '/g/11f6154bys',
'Night at the Museum': '/m/0bmssv',
'Fantastic 4: Rise of the Silver Surfer': '/m/09sh8k',
'Wrath of the Titans': '/m/0cs17mf',
'Safe House': '/m/0fq2d51',
```

```
'Guardians of the Galaxy Vol. 2': '/m/011cjkjg',
'Finding Dory': '/m/0sghvky',
'Pink Panther': '/m/0dlbh',
'Fifty Shades Freed': '/g/11c3x8gfn9',
'The Magnificent Seven': '/g/11b7k3tft3',
'The Lion King': '/m/0m63c',
'Harry Potter and the Goblet of Fire': '/m/031786',
'Twilight': '/m/03nm_fh',
'Girls Trip': '/g/11c0pzv7fl',
'Mission: Impossible - Rogue Nation': '/m/0w0qchd',
'Zombieland': '/m/0cj0v2b',
'Mean Girls': '/m/0ch4824',
'Toy Story 4': '/g/122xf7xr',
'Coach Carter': '/m/04xx9s',
'The Exorcism of Emily Rose': '/m/07ntww',
'Kicking & Screaming': '/m/05spvb',
'Alice Through the Looking Glass': '/m/0y4m_xb',
"Don't Breathe": '/g/11btqnjr6n',
'Mamma Mia!': '/m/0272_vz',
"Madea's Witness Protection": '/m/0h_csjp',
'Shrek 2': '/m/02lk60',
'Avengers: Age of Ultron': '/m/0n15g8q',
'Ready Player One': '/g/11bxc649kh',
'Act of Valor': '/m/0gwrh1q',
'The Adjustment Bureau': '/m/06zjsc_',
'Four Brothers': '/m/07334n',
'21 Jump Street': '/m/0bq8tmw',
'Think Like A Man': '/m/0gwf191',
'Neighbors 2: Sorority Rising': '/g/11c3mqbys6',
'The Day the Earth Stood Still': '/m/03gvywx',
'Julie & Julia': '/m/0416y94',
'Resident Evil: Retribution': '/m/0gtv7pk',
'Ant-Man and the Wasp': '/g/11bw82pznj',
'The Grinch': '/g/11bz0yq44d',
'Safe Haven': '/m/0gwgmd3',
'Alvin and the Chipmunks: Chipwrecked': '/m/0h1cdwq',
'Shark Tale': '/m/01xbxn',
'22 Jump Street': '/m/0v_w91n',
'Prince of Persia: The Sands of Time': '/m/047gn4y',
'Shrek the Third': '/m/03nfnx',
'TMNT': '/m/0j7l262',
'Dodgeball': '/m/034qrh',
'Evan Almighty': '/m/07p62k',
'Paranormal Activity 4': '/m/0hyp0b8',
'The Lego Ninjago Movie': '/g/11cjh_4ptv',
'Bridesmaids': '/m/0ds35l9',
'Creed': '/g/11c2r14f_q',
'The Incredible Hulk': '/m/0dnkmq',
'It': '/g/11f4qykqpj',
'Alvin and the Chipmunks': '/m/02r4y3t',
'The Final Destination': '/m/03qdlh6',
"Ocean's Eight": '/g/11c1c8mc72',
'After Earth': '/m/0hhgh70',
'Cloverfield': '/g/11bzx49rn0',
'Alien vs. Predator': '/m/03n785',
'Fantastic Four': '/m/02rx09l',
'Fast Five': '/m/0bq6ntw',
"Pirates of the Caribbean: Dead Man's Chest": '/m/027j3vt',
'Life of Pi': '/m/0fpv_3_',
"Daddy's Home": '/m/0wybbd2',
'The Dukes of Hazzard': '/m/033f8n',
'Project X': '/m/0gg74r6',
'Iron Man 2': '/m/05qbckf',
'Lucy': '/m/0w36vqd',
'Prom Night': '/m/02rxbq6',
```

```
'Unstoppable': '/m/076zy_g',
'The Conjuring 2': '/g/1pwf7_9rd',
'Crazy Rich Asians': '/g/11g71c9nmg',
'Jarhead': '/g/11f0wklmz1',
'Logan': '/g/11cn3h3kmj',
'Kingsman: The Golden Circle': '/g/11by_n1rp1',
'The Fault in Our Stars': '/m/0wyjt0p',
'The Lego Movie': '/m/0n4g1dj',
'Peter Rabbit': '/g/11c619qzm8',
'Dawn of the Planet of the Apes': '/m/0n05bkp'}
```

In [193…

```
# Pulling data from pytrends

# Exclude movies without found film keyword
movie_names = list(search_dict.keys())
df_has_endgame = False
start = 0
step = 4  # maximum number of requests build_payload can handle at a time - 1

movie_interest_df = pd.DataFrame(columns=['dummy'], index=range(0))

# store to movie_interest_df the interest in movies in groups of 4 at a time
while start <= len(movie_names):
    # adding Avengers: Endgame (movie with top box office earnings) to each subset in o
    movie_subset = list(set(movie_names[start:start+step] + ['Avengers: Endgame']))
    key_subset = [search_dict[x] for x in movie_subset]
    pytrends.build_payload(key_subset, timeframe='all', geo='US', gprop='')

    got_dataframe = False
    while not got_dataframe:
        try:
            temp_df = pytrends.interest_over_time()
            got_dataframe = True
        except:
            print("Read timeout . . .")
            time.sleep(3)
    temp_df.columns = movie_subset + ['isPartial']
    if not temp_df.empty:
        if df_has_endgame:
            temp_df = temp_df.drop(labels=['isPartial', 'Avengers: Endgame'],axis='colu
        else:
            temp_df = temp_df.drop(labels=['isPartial'],axis='columns')
            df_has_endgame = True
        movie_interest_df = pd.concat([movie_interest_df, temp_df], axis=1)
    start += step

    # status messages
    if start >= len(movie_names):
        print("Done! Added data for all", len(movie_names), "movies.")
    if (start + step) % 100 == 0:
        print("Added data for", start + step, "out of", len(movie_names), "movies.")
    if (start + step) % 300 == 0:
        print("Waiting to prevent error 429 (too many requests)...")
        time.sleep(30)
```

```
Added data for 100 out of 730 movies.
Added data for 200 out of 730 movies.
Added data for 300 out of 730 movies.
Waiting to prevent error 429 (too many requests)...
Added data for 400 out of 730 movies.
```

```
Added data for 500 out of 730 movies.
Added data for 600 out of 730 movies.
Waiting to prevent error 429 (too many requests)...
Added data for 700 out of 730 movies.
Done! Added data for all 730 movies.
```

In [194…

```
# drop temporary column and transpose movie_interest_df to match opening_weekend_df
movie_interest_df = movie_interest_df.drop(labels=['dummy'], axis='columns', errors='ig
movie_interest_df = movie_interest_df.transpose()

# check data shape
assert movie_interest_df.shape == (len(movie_names), 206)
movie_interest_df.head(len(movie_names))
```

Out[194…

	2004-01-01	2004-02-01	2004-03-01	2004-04-01	2004-05-01	2004-06-01	2004-07-01	2004-08-01	2004-09-01	2004-10-01	…	2020-05-01	2020-06-01
Vantage Point	0	0	0	0	0	0	0	0	0	0	…	0	0
Cars	0	0	0	0	0	0	0	0	0	0	…	1	1
Big Hero 6	0	0	0	0	0	0	0	0	0	0	…	1	1
Avengers: Endgame	0	0	0	0	0	0	0	0	0	0	…	5	4
xXx: Return of Xander Cage	0	0	0	0	0	0	0	0	0	0	…	0	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…
Kingsman: The Golden Circle	0	0	0	0	0	0	0	0	0	0	…	0	0
The Fault in Our Stars	0	0	0	0	0	0	0	0	0	0	…	0	0
The Lego Movie	0	0	0	0	0	0	0	0	0	0	…	1	0
Peter Rabbit	0	0	0	0	0	0	0	0	0	0	…	0	0
Dawn of the Planet of the Apes	0	0	0	0	0	0	0	0	0	0	…	0	0

730 rows × 206 columns

In [195…

```
# Filter out movies that didn't have film search keywords
movie_names = list(search_dict.keys())
opening_weekend_df = opening_weekend_df[opening_weekend_df.name.isin(movie_names)]
```

```
# Each entry in opening_weekend_df has an entry in movie_interest_df and vice versa
assert all(opening_weekend_df.name.isin(movie_interest_df.index))
assert all(movie_interest_df.index.isin(opening_weekend_df.name))

# Save data
movie_interest_df.to_csv('data/movie_interest.csv')
opening_weekend_df.to_csv('data/opening_weekend.csv')
```

Data Visualization:

We first looked at the data type objects of `opening_weekend_df` and `movie_interest_df`, and we also used `.describe()` to view the summary statistics of both dataframes. We decided to visualize the frequency of opening weekend earnings through a barplot, which shows around $25,000,000 as the most common amount a movie usually earns during their opening weekend. After, we found the upper and lower bounds of the opening weekend revenue variable `opening_weekend_df['opening']` and deemed outliers as observations higher than the `upper_bound` value. We then dropped all the outliers from `opening_weekend_df` and put the remaining observations in `no_outliers_df`.

In [198…

```
# Loading data
opening_weekend_df = pd.read_csv('data/opening_weekend.csv',index_col=0)
movie_interest_df = pd.read_csv('data/movie_interest.csv')

# Format columns of movie interest dataframe to make it easier to work with
new_columns = []
for i in range(len(movie_interest_df.columns)):
    if i == 0:
        new_columns.append("movie")
    else:
        new_column = movie_interest_df.columns[i].split(' ')[0]
        new_columns.append(new_column)

movie_interest_df.columns = new_columns
movie_interest_df = movie_interest_df.set_index('movie')
```

Taking a closer look at our top box office movies:

In [199…

```
opening_weekend_df.head()
```

Out[199…

	rank	name	opening	total	open_percent	theaters	average	date	dist
0	1	Avengers: Endgame	357115007	858373000	41.6	4662	76601	2019-04-26	Walt Disney Studios Motion Pictures
1	2	Avengers: Infinity War	257698183	678815482	38.0	4474	57599	2018-04-27	Walt Disney Studios Motion Pictures

	rank	name	opening	total	open_percent	theaters	average	date	dist
2	3	Star Wars: Episode VII - The Force Awakens	247966675	936662225	26.5	4134	59982	2015-12-18	Walt Disney Studios Motion Pictures
3	4	Star Wars: Episode VIII - The Last Jedi	220009584	620181382	35.5	4232	51987	2017-12-15	Walt Disney Studios Motion Pictures
4	5	Jurassic World	208806270	652270625	32.0	4274	48855	2015-06-12	Universal Pictures

In [200…
```
# confirming that there are not any null values in the opening weekend dataset
assert not any(opening_weekend_df.isnull().any(axis=1))
```

In [201…
```
print("Shape of opening_weekend:", opening_weekend_df.shape)
```

```
Shape of opening_weekend: (764, 9)
```

In [202…
```
opening_weekend_df.dtypes
```

Out[202…
```
rank             int64
name            object
opening          int64
total            int64
open_percent   float64
theaters         int64
average          int64
date            object
dist            object
dtype: object
```

In [203…
```
opening_weekend_df.describe()
```

Out[203…

	rank	opening	total	open_percent	theaters	average
count	764.000000	7.640000e+02	7.640000e+02	764.000000	764.000000	764.000000
mean	477.540576	4.677196e+07	1.472382e+08	33.412565	3469.558901	12975.467277
std	292.437866	3.652994e+07	1.118747e+08	8.496575	563.734964	8240.262786
min	1.000000	1.981058e+07	3.434194e+07	8.900000	683.000000	4859.000000
25%	215.750000	2.472484e+07	7.526681e+07	27.800000	3109.500000	7871.000000
50%	470.500000	3.352897e+07	1.106705e+08	33.100000	3498.000000	10357.000000
75%	731.250000	5.432274e+07	1.766809e+08	38.900000	3898.500000	14483.000000
max	999.000000	3.571150e+08	9.366622e+08	63.600000	4725.000000	76601.000000

Taking a closer look at the interest in our top box office movies:

In [204…
```
movie_interest_df.head()
```

Out[204…

movie	2004-01-01	2004-02-01	2004-03-01	2004-04-01	2004-05-01	2004-06-01	2004-07-01	2004-08-01	2004-09-01	2004-10-01	...	2020-05-01	2020-06-01
Vantage Point	0	0	0	0	0	0	0	0	0	0	...	0	0
Cars	0	0	0	0	0	0	0	0	0	0	...	1	1
Big Hero 6	0	0	0	0	0	0	0	0	0	0	...	1	1
Avengers: Endgame	0	0	0	0	0	0	0	0	0	0	...	5	4
xXx: Return of Xander Cage	0	0	0	0	0	0	0	0	0	0	...	0	0

5 rows × 206 columns

In [205…
```
# confirming that there are not any null values in the movie_interest dataset
assert not any(movie_interest_df.isnull().any(axis=1))
```

In [206…
```
print("Shape of movie_interest:", movie_interest_df.shape)
```

```
Shape of movie_interest: (730, 206)
```

In [207…
```
movie_interest_df.dtypes
```

Out[207…
```
2004-01-01    int64
2004-02-01    int64
2004-03-01    int64
2004-04-01    int64
2004-05-01    int64
              ...
2020-10-01    int64
2020-11-01    int64
2020-12-01    int64
2021-01-01    int64
2021-02-01    int64
Length: 206, dtype: object
```

In [208…
```
movie_interest_df.describe()
```

Out[208…

	2004-01-01	2004-02-01	2004-03-01	2004-04-01	2004-05-01	2004-06-01	2004-07-01	2004-08-01

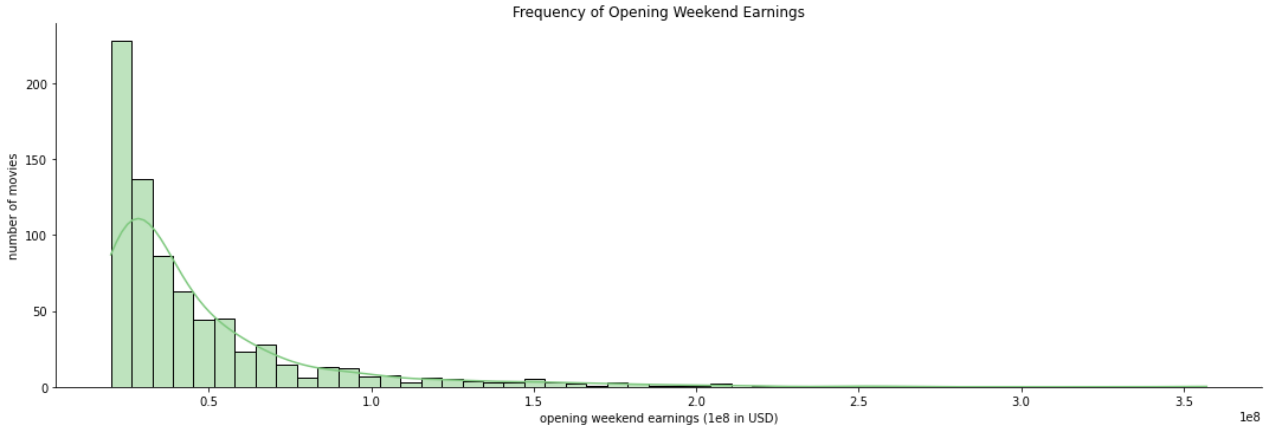
	2004-01-01	2004-02-01	2004-03-01	2004-04-01	2004-05-01	2004-06-01	2004-07-01	2004-08-01
count	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000	730.000000
mean	0.134247	0.163014	0.165753	0.160274	0.204110	0.208219	0.227397	0.176712
std	0.646658	1.656182	1.174106	0.879774	1.107765	0.918004	1.100101	0.744623
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	10.000000	43.000000	27.000000	16.000000	19.000000	11.000000	18.000000	12.000000

8 rows × 206 columns

Removing outliers for top box office earning movies:

In [209…

```
# distribution of opening weekend earnings before outlier removal
sns.set_palette(sns.color_palette("Accent", 1))
ax = sns.displot(opening_weekend_df['opening'], kde=True, height=5, aspect=3)

ax.set_axis_labels("opening weekend earnings (1e8 in USD)", "number of movies")
ax.set(title="Frequency of Opening Weekend Earnings")
plt.show()
```



In [210…

```
# determining quartiles
q1 = opening_weekend_df['opening'].describe()[4]
q3 = opening_weekend_df['opening'].describe()[6]
iqr = q3-q1
lower_bound = q1 - 1.5*iqr
upper_bound = q3 + 1.5*iqr
lower_bound, upper_bound
```

Out[210…

```
(-19672020.125, 98719598.875)
```

In [211…
```
# Looking at the movies with revenues above the upper bound
outliers = opening_weekend_df[opening_weekend_df['opening'] > upper_bound]
outliers
```

Out[211…

	rank	name	opening	total	open_percent	theaters	average	date	dist
0	1	Avengers: Endgame	357115007	858373000	41.6	4662	76601	2019-04-26	Walt Disney Studios Motion Pictures
1	2	Avengers: Infinity War	257698183	678815482	38.0	4474	57599	2018-04-27	Walt Disney Studios Motion Pictures
2	3	Star Wars: Episode VII - The Force Awakens	247966675	936662225	26.5	4134	59982	2015-12-18	Walt Disney Studios Motion Pictures
3	4	Star Wars: Episode VIII - The Last Jedi	220009584	620181382	35.5	4232	51987	2017-12-15	Walt Disney Studios Motion Pictures
4	5	Jurassic World	208806270	652270625	32.0	4274	48855	2015-06-12	Universal Pictures
5	6	The Avengers	207438708	623357910	33.3	4349	47698	2012-05-04	Walt Disney Studios Motion Pictures
6	7	Black Panther	202003951	700059566	28.9	4020	50249	2018-02-16	Walt Disney Studios Motion Pictures
7	8	The Lion King	191770759	543638043	35.3	4725	40586	2019-07-19	Walt Disney Studios Motion Pictures
8	9	Avengers: Age of Ultron	191271109	459005868	41.7	4276	44731	2015-05-01	Walt Disney Studios Motion Pictures
9	10	Incredibles 2	182687905	608581744	30.0	4410	41425	2018-06-15	Walt Disney Studios Motion Pictures
10	11	Captain America: Civil War	179139142	408084349	43.9	4226	42389	2016-05-06	Walt Disney Studios Motion Pictures
11	12	Star Wars: Episode IX - The Rise of Skywalker	177383864	515202542	34.4	4406	40259	2019-12-20	Walt Disney Studios Motion Pictures

	rank	name	opening	total	open_percent	theaters	average	date	dist
12	13	Beauty and the Beast	174750616	504014165	34.7	4210	41508	2017-03-17	Walt Disney Studios Motion Pictures
13	14	Iron Man 3	174144585	409013994	42.6	4253	40946	2013-05-03	Walt Disney Studios Motion Pictures
14	15	Harry Potter and the Deathly Hallows: Part 2	169189427	381011219	44.4	4375	38671	2011-07-15	Warner Bros.
15	16	Batman v Superman: Dawn of Justice	166007347	330360194	50.2	4242	39134	2016-03-25	Warner Bros.
16	17	The Dark Knight Rises	160887295	448139099	35.9	4404	36532	2012-07-20	Warner Bros.
17	18	The Dark Knight	158411483	533345358	29.7	4366	36282	2008-07-18	Warner Bros.
18	19	The Hunger Games: Catching Fire	158074286	424668047	37.2	4163	37971	2013-11-22	Lionsgate
19	20	Rogue One: A Star Wars Story	155081681	532177324	29.1	4157	37306	2016-12-16	Walt Disney Studios Motion Pictures
20	21	Captain Marvel	153433423	426829839	36.0	4310	35599	2019-03-08	Walt Disney Studios Motion Pictures
21	22	The Hunger Games	152535747	408010692	37.4	4137	36871	2012-03-23	Lionsgate
22	23	Spider-Man 3	151116516	336530303	44.9	4252	35540	2007-05-04	Sony Pictures Entertainment (SPE)
23	24	Jurassic World: Fallen Kingdom	148024610	417719760	35.4	4475	33078	2018-06-22	Universal Pictures
24	25	Furious 7	147187040	353007020	41.7	4004	36760	2015-04-03	Universal Pictures
25	26	Guardians of the Galaxy Vol. 2	146510104	389813101	37.6	4347	33703	2017-05-05	Walt Disney Studios Motion Pictures

	rank	name	opening	total	open_percent	theaters	average	date	dist
26	27	The Twilight Saga: New Moon	142839137	296623634	48.2	4024	35496	2009-11-20	Summit Entertainment
27	28	The Twilight Saga: Breaking Dawn - Part 2	141067634	292324737	48.3	4070	34660	2012-11-16	Lionsgate
28	29	The Twilight Saga: Breaking Dawn - Part 1	138122261	281287133	49.1	4061	34011	2011-11-18	Summit Entertainment
29	30	Pirates of the Caribbean: Dead Man's Chest	135634554	423315812	32.0	4133	32817	2006-07-07	Walt Disney Studios Motion Pictures
30	31	Finding Dory	135060273	486295561	27.8	4305	31372	2016-06-17	Walt Disney Studios Motion Pictures
31	32	Suicide Squad	133682248	325100054	41.1	4255	31417	2016-08-05	Warner Bros.
32	33	Deadpool	132434639	363070709	36.5	3558	37221	2016-02-12	Twentieth Century Fox
33	34	Frozen II	130263358	477373578	27.3	4440	29338	2019-11-22	Walt Disney Studios Motion Pictures
34	35	Iron Man 2	128122480	312433331	41.0	4380	29251	2010-05-07	Paramount Pictures
35	36	Deadpool 2	125507153	318491426	39.4	4349	28858	2018-05-18	Twentieth Century Fox
36	37	Harry Potter and the Deathly Hallows: Part 1	125017372	295983305	42.2	4125	30307	2010-11-19	Warner Bros.
37	38	It	123403419	327481748	37.7	4103	30076	2017-09-08	Warner Bros.
38	39	Thor: Ragnarok	122744989	315058289	39.0	4080	30084	2017-11-03	Walt Disney Studios Motion Pictures
39	40	The Hunger Games: Mockingjay - Part 1	121897634	337135885	36.2	4151	29365	2014-11-21	Lionsgate
40	41	Shrek the Third	121629270	322719944	37.7	4122	29507	2007-05-18	DreamWorks

	rank	name	opening	total	open_percent	theaters	average	date	dist
41	42	Toy Story 4	120908065	434038008	27.9	4575	26427	2019-06-21	Walt Disney Studios Motion Pictures
42	43	Spider-Man: Homecoming	117027503	334201140	35.0	4348	26915	2017-07-07	Sony Pictures Entertainment (SPE)
43	44	Man of Steel	116619362	291045518	40.1	4207	27720	2013-06-14	Warner Bros.
44	45	Alice in Wonderland	116101023	334191110	34.7	3728	31142	2010-03-05	Walt Disney Studios Motion Pictures
45	46	Minions	115718405	336045770	34.4	4301	26905	2015-07-10	Universal Pictures
47	48	Pirates of the Caribbean: At World's End	114732820	309420425	37.1	4362	26302	2007-05-25	Walt Disney Studios Motion Pictures
48	49	Toy Story 3	110307189	415004880	26.6	4028	27385	2010-06-18	Walt Disney Studios Motion Pictures
49	50	Transformers: Revenge of the Fallen	108966307	402111870	27.1	4234	25736	2009-06-24	DreamWorks
50	51	Star Wars: Episode III - Revenge of the Sith	108435841	380270577	28.5	3661	29619	2005-05-19	Twentieth Century Fox
51	52	Shrek 2	108037878	441226247	24.5	4163	25951	2004-05-19	DreamWorks Distribution
52	53	The Secret Life of Pets	104352905	368384330	28.3	4370	23879	2016-07-08	Universal Pictures
53	54	The Jungle Book	103261464	364001123	28.4	4028	25635	2016-04-15	Walt Disney Studios Motion Pictures
54	55	Wonder Woman	103251471	412563408	25.0	4165	24790	2017-06-02	Warner Bros.
55	56	X-Men: The Last Stand	102750665	234362462	43.8	3690	27845	2006-05-26	Twentieth Century Fox
56	57	Harry Potter and the Goblet of Fire	102685961	290013036	35.4	3858	26616	2005-11-18	Warner Bros.

	rank	name	opening	total	open_percent	theaters	average	date	dist
57	58	The Hunger Games: Mockingjay - Part 2	102665981	281723902	36.4	4175	24590	2015-11-20	Lionsgate
58	59	Indiana Jones and the Kingdom of the Crystal S...	100137835	317101119	31.6	4260	23506	2008-05-22	Paramount Pictures
59	60	Transformers: Age of Extinction	100038390	245439076	40.8	4233	23632	2014-06-27	Paramount Pictures
60	61	The Fate of the Furious	98786705	226008385	43.7	4310	22920	2017-04-14	Universal Pictures

In [212…

```
# removing all outliers from dataset
no_outliers_df = opening_weekend_df.copy()
outlier_list = outliers.name.tolist()
outlier_list

for movie in outlier_list:
    indices = no_outliers_df[no_outliers_df['name'] == movie].index
    no_outliers_df.drop(indices, inplace = True)
```

In [213…

```
# THIS DATAFRAME HAS NO OUTLIERS, USE OPENING_WEEKEND_DF TO INCLUDE OUTLIERS
no_outliers_df.head()
```

Out[213…

	rank	name	opening	total	open_percent	theaters	average	date	dist
61	62	Iron Man	98618668	318604126	31.0	4105	24024	2008-05-02	Paramount Pictures
62	63	Transformers: Dark of the Moon	97852865	352390543	27.8	4088	23936	2011-06-29	DreamWorks
63	64	Fast & Furious 6	97375245	238679850	40.8	3658	26619	2013-05-24	Universal Pictures
64	65	Joker	96202337	335451311	28.7	4374	21994	2019-10-04	Warner Bros.
65	66	Captain America: The Winter Soldier	95023721	259766572	36.6	3938	24129	2014-04-04	Walt Disney Studios Motion Pictures

In [214…

```
print(no_outliers_df.describe())
no_outliers_df.to_csv('data/no_outliers.csv')
```

```
              rank       opening         total  open_percent     theaters  \
count   704.000000  7.040000e+02  7.040000e+02    704.000000   704.000000
```

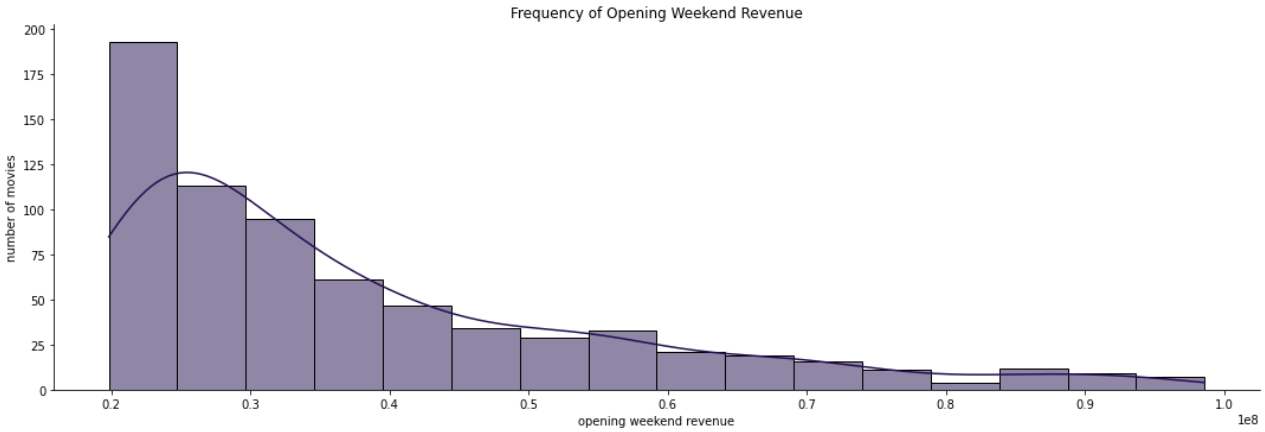
```
mean    515.620739  3.818596e+07  1.241467e+08     33.176847  3406.261364
std     272.583044  1.810523e+07  7.117028e+07      8.611834   538.345838
min      62.000000  1.981058e+07  3.434194e+07      8.900000   683.000000
25%     273.750000  2.438911e+07  7.311826e+07     27.500000  3082.750000
50%     511.000000  3.155911e+07  1.026592e+08     32.700000  3439.000000
75%     749.250000  4.703110e+07  1.583017e+08     38.700000  3777.250000
max     999.000000  9.861867e+07  7.497661e+08     63.600000  4634.000000
```

```
            average
count     704.000000
mean    11107.528409
std      4553.461177
min      4859.000000
25%      7732.500000
50%      9949.000000
75%     13210.750000
max     45560.000000
```

The mean recorded earlier for all movies' opening weekend revenue was 4.725279e+07. After removing outliers, we observe that the center of the data has shifted to a lower amount between 3e+07 and 4e+07.

In [215…

```
# distribution of opening weekend revenue without outliers
sns.set_palette(sns.color_palette("magma"),1)
ax = sns.displot(no_outliers_df['opening'], kde=True, height=5, aspect=3)

ax.set(title="Frequency of Opening Weekend Revenue")
ax.set_axis_labels("opening weekend revenue", "number of movies")
plt.show()
```



Using `no_outliers_df` , we plotted the frequency of the opening weekend earnings again, and the resulting histogram (in purple) shows a less extreme right skew in comparison to the previous histogram (in green). Without outliers, the average amount of revenue a movie earns during their opening weekend is around $38,000,000 which is about \$8,000,000 less than the average amount of revenue of the movies in the original dataset.

Studying the relationship between Movie Distributors and Total Opening Weekend Revenue

After, we decided to study the relationship between movie distributors and the total opening weekend revenue in order to see which movie distributor amassed a larger amount of opening

weekend revenue. We first grouped all the opening weekend revenues by the distributors using `groupby`, and we placed the resulting data in `dist_revenue_df`.

In [216...
```
# group the movies by their shared distributors and find the total sum of their opening
grouped_dist_df = no_outliers_df.groupby(["dist"]).sum().reset_index()
dist_revenue_df = grouped_dist_df[['dist', 'opening']]
dist_revenue_df = dist_revenue_df.sort_values(by='opening', ascending=False).reset_inde
dist_revenue_df.describe()
```

Out[216...

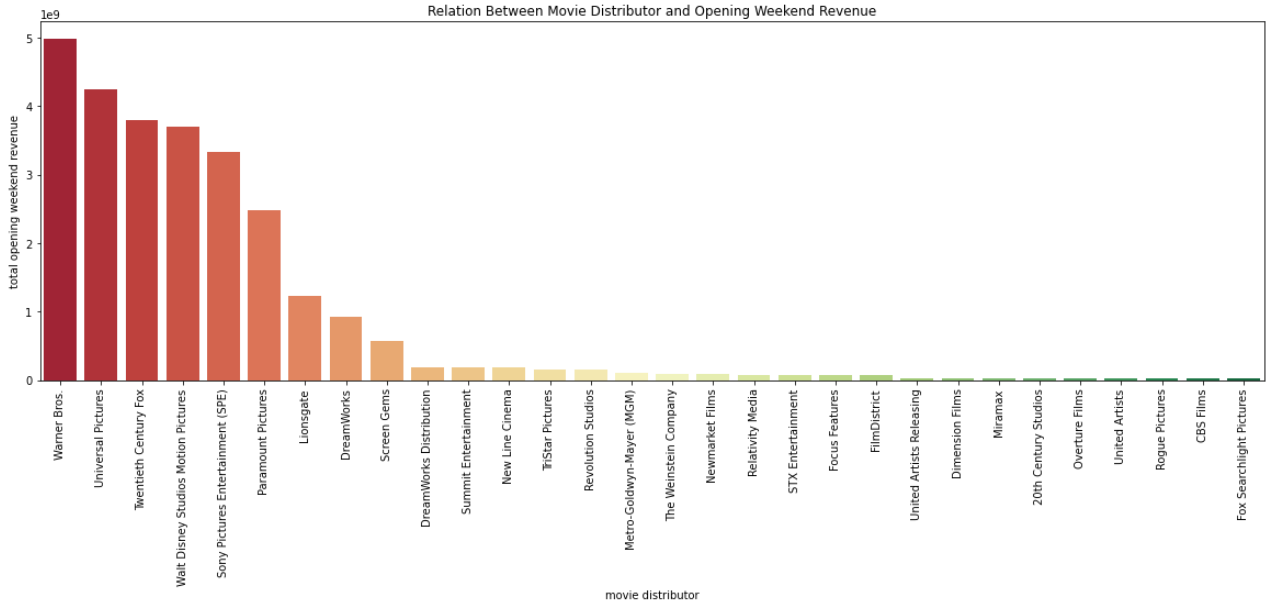
	opening
count	3.000000e+01
mean	8.960972e+08
std	1.518259e+09
min	2.049760e+07
25%	2.865000e+07
50%	9.979421e+07
75%	8.334361e+08
max	4.982723e+09

In [217...
```
dist_revenue_df.head()
```

Out[217...

	dist	opening
0	Warner Bros.	4982722638
1	Universal Pictures	4238271550
2	Twentieth Century Fox	3791487942
3	Walt Disney Studios Motion Pictures	3695427034
4	Sony Pictures Entertainment (SPE)	3329397074

In [218...
```
# total opening weekend revenues by movie distributor

plt.figure(figsize=(20,6))
ax = sns.barplot(x="dist", y="opening", data=dist_revenue_df, palette=sns.color_palette
ax.set(title="Relation Between Movie Distributor and Opening Weekend Revenue")
ax.set_xticklabels(dist_revenue_df['dist'], rotation = 90)
ax.set(xlabel='movie distributor', ylabel='total opening weekend revenue')
plt.show()
```

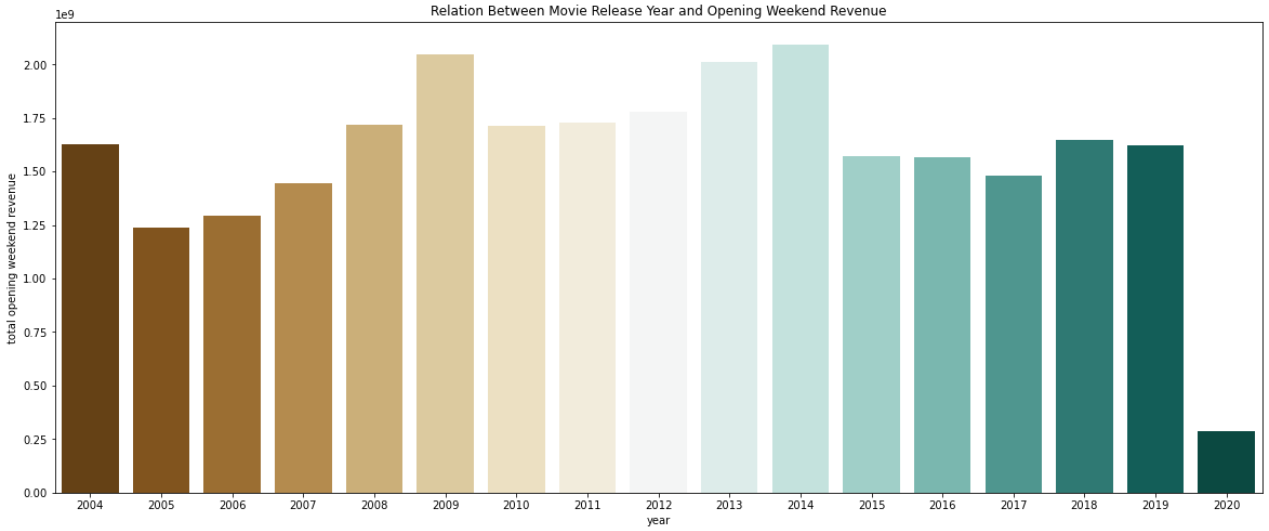


Unsurprisingly, Warner Bros. comes out at top with opening revenue while Universal Pictures has the second highest total opening weekend revenue. Note that out of the 30 distributors here, only 8 of them have total earnings the mean: 8.580822e+08. This shows that the more well-known distributors tend to garner a higher opening weekend revenue in comparison to lesser known distributors. More revenue could mean a higher budget for future films, which could then lead to higher impact marketing and an increase in movie searches.

Studying the relationship between Year and Total Opening Weekend Revenue

In [219…

```
# observing how total opening weekend revenue changed from 2004 to 2020

# tailoring dataset by grouping movies by the year of their release and find the total
year_df = no_outliers_df
year_df[['year','month','day']] = year_df.date.str.split("-", expand=True)
grouped_year_df = year_df.groupby(['year']).sum().reset_index()
year_revenue_df = grouped_year_df[['year', 'opening']]
year_revenue_df.head()

# plot configurations
plt.figure(figsize=(20,8))
ax = sns.barplot(x="year", y="opening", palette=sns.color_palette("BrBG", 17), data=yea
ax.set(title="Relation Between Movie Release Year and Opening Weekend Revenue")
ax.set(xlabel='year', ylabel='total opening weekend revenue')
plt.show()
```



In [220…

```
# finding number of movies in list made in 2020
movies_in_2020 = year_df[year_df['year'] == '2020']
print("Number of top 1000 box office earnings movies in 2020:", len(movies_in_2020))
movies_in_2020
```

```
Number of top 1000 box office earnings movies in 2020: 8
```

Out[220…

	rank	name	opening	total	open_percent	theaters	average	date	dist	yea
159	160	Bad Boys for Life	62504105	206305244	30.3	3775	16557	2020-01-17	Sony Pictures Entertainment (SPE)	202
180	181	Sonic the Hedgehog	58018348	148974665	39.0	4167	13923	2020-02-14	Paramount Pictures	202
373	374	Onward	39119861	61555145	63.6	4310	9076	2020-03-06	Walt Disney Studios Motion Pictures	202
484	485	Birds of Prey	33010017	84158461	39.2	4236	7792	2020-02-07	Warner Bros.	202
613	614	The Invisible Man	28205665	70410000	40.1	3610	7813	2020-02-28	Universal Pictures	202
726	727	The Call of the Wild	24791624	62342368	39.8	3752	6607	2020-02-21	20th Century Studios	202
869	870	Dolittle	21844045	77047065	28.4	4155	5257	2020-01-17	Universal Pictures	202
972	973	Tenet	20200000	57929000	34.9	2810	7188	2020-09-03	Warner Bros.	202

In [221…

```
# finding number stats on # of movies made per year
print(year_df['year'].value_counts())
print()
print("Average number of top box office earning movies from 2004-2019:", np.mean(year_d
```

```
2009    54
2013    52
2010    49
2012    48
2014    48
2008    46
2004    45
2016    44
2011    43
2006    41
2018    40
2019    39
2017    37
2015    37
2005    37
2007    36
2020     8
Name: year, dtype: int64
```

```
Average number of top box office earning movies from 2004-2019: 43.5
```

2020 appears to be doing rather poorly compared to previous years. A closer look at the data tells us that only 8 movies from 2020 made it to the top 1000 box office earners list. The average number of movies to make the list per each year is 43 and the range is 18.

Checking for outliers in movie interest data:

We checked for outliers using the code below. We looked for max search popularities that exceeded Avenger's Endgame. We found 200 movies that exceeded Avengers: Endgame, but on manually checking with Google Trends, we found that many of these movies did not actually exceed the popularity of Avengers: Endgame. Instead of finding outliers, we found that the way we collected data for search trends was incorrect. To fix it, we used Google suggestions to narrow down our search to be specific to film searches of our movies, not just any search. Upon doing this, we fixed the errors in our data. The changes can be found in the data cleaning section, in the code starting with comment "Pulling search keywords from pytrends."

In [222…

```
# Bits of code we wrote finding that our data was incorrect
test = movie_interest_df.transpose().describe()
test = test.transpose()
# Displayed all movies with higher search popularity.
# Will display different output after data was fixed
# 46 was used because Avengers: Endgame had max search popularity of 46
test[test['max'] > 46]
```

Out[222…

movie	count	mean	std	min	25%	50%	75%	max
Avengers: Endgame	206.0	3.480583	9.013113	0.0	0.0	2.0	3.0	100.0
Fifty Shades of Grey	206.0	2.504854	5.615091	0.0	0.0	1.0	3.0	60.0
The Dark Knight	206.0	2.359223	4.866720	0.0	1.0	1.0	2.0	55.0
The Hunger Games: Catching Fire	206.0	3.461165	7.046601	0.0	0.0	2.0	4.0	79.0
The Incredibles	206.0	1.815534	3.891824	0.0	1.0	1.0	2.0	51.0

movie	count	mean	std	min	25%	50%	75%	max
Star Wars: Episode III - Revenge of the Sith	206.0	2.359223	4.550753	1.0	1.0	1.0	2.0	48.0
Deadpool	206.0	2.111650	4.879240	0.0	0.0	1.0	2.0	53.0
Rogue One: A Star Wars Story	206.0	0.771845	5.223902	0.0	0.0	0.0	0.0	73.0
Jurassic World	206.0	1.334951	4.195170	0.0	0.0	0.0	1.0	50.0
Star Wars: Episode VIII - The Last Jedi	206.0	1.563107	4.124099	0.0	0.0	1.0	2.0	49.0
Black Panther	206.0	1.378641	6.151925	0.0	0.0	0.0	1.0	77.0
Batman Begins	206.0	2.359223	4.866720	0.0	1.0	1.0	2.0	55.0
The Avengers	206.0	4.490291	8.339585	0.0	0.0	3.0	5.0	69.0
Avengers: Infinity War	206.0	1.665049	5.298821	0.0	0.0	0.0	1.0	50.0
Star Wars: Episode VII - The Force Awakens	206.0	3.082524	6.170582	1.0	1.0	2.0	3.0	76.0
The Dark Knight Rises	206.0	1.092233	3.776208	0.0	0.0	0.0	1.0	48.0
Suicide Squad	206.0	1.189320	4.318772	0.0	0.0	0.0	1.0	55.0
Twilight	206.0	6.631068	9.860975	0.0	2.0	3.0	6.0	81.0
It	206.0	1.689320	4.502637	0.0	1.0	1.0	1.0	53.0

After fixing our data, only Avengers: Endgame had max of 100, as we expected, based on the virality of the movie and its general popularity in internet culture. To check for further anomalies, we selected movies from the data at random and manually checked them with Google Trends.

In [223…
```
# After fixing our data, we found the following
test = movie_interest_df.transpose().describe().transpose()
test[test['max']==100]
```

Out[223…

movie	count	mean	std	min	25%	50%	75%	max
Avengers: Endgame	206.0	3.480583	9.013113	0.0	0.0	2.0	3.0	100.0

Unfortunately, our fix had some flaws. Some movies were conflated with other, less popular search terms, resulting in some movies having max search popularity of zero. Others were searched too little to give us significant data. All of these movies can be seen below.

In [224…
```
test[test['max'] == 0]
```

Out[224…

movie	count	mean	std	min	25%	50%	75%	max
The Other Woman	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
White Noise	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

movie	count	mean	std	min	25%	50%	75%	max
The Strangers	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Gravity	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Super 8	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Beverly Hills Chihuahua	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Sing	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Us	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Marley & Me	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Neighbors	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Bewitched	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Notorious	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
The Perfect Guy	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Tomb Raider	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
The Longest Yard	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Prisoners	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Beowulf	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Big Momma's House 2	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Oblivion	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Dark Shadows	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Rocketman	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Fantastic Beasts and Where to Find Them	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Cinderella	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
The Interpreter	206.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

So, we decided to drop these movies from our dataset.

In [225…
```
movie_interest_df = movie_interest_df.drop(test[test['max'] == 0].index)
```

In [226…
```
movie_interest_df.head()
```

Out[226…

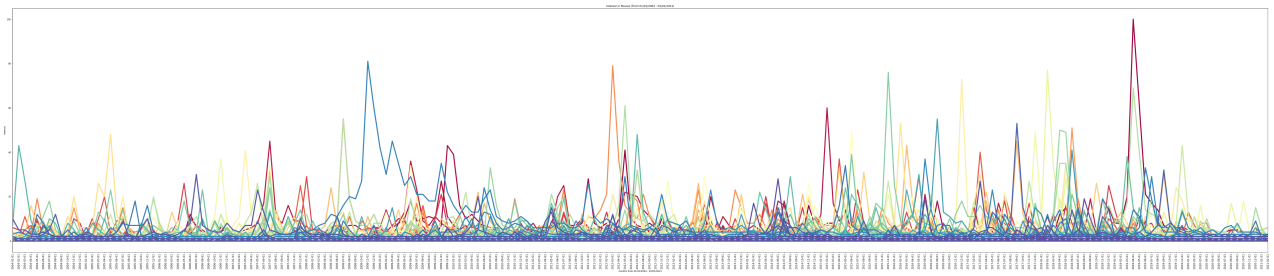
movie	2004-01-01	2004-02-01	2004-03-01	2004-04-01	2004-05-01	2004-06-01	2004-07-01	2004-08-01	2004-09-01	2004-10-01	...	2020-05-01	2020-06-01
Vantage Point	0	0	0	0	0	0	0	0	0	0	...	0	0
Cars	0	0	0	0	0	0	0	0	0	0	...	1	1

movie	2004-01-01	2004-02-01	2004-03-01	2004-04-01	2004-05-01	2004-06-01	2004-07-01	2004-08-01	2004-09-01	2004-10-01	...	2020-05-01	2020-06-01
Big Hero 6	0	0	0	0	0	0	0	0	0	0	...	1	1
Avengers: Endgame	0	0	0	0	0	0	0	0	0	0	...	5	4
xXx: Return of Xander Cage	0	0	0	0	0	0	0	0	0	0	...	0	0

5 rows × 206 columns

In [227…

```
# observing how interest in a movie changed from 2004-2020 by month

total_interest = movie_interest_df.transpose()

# plot configurations
cmap = plt.get_cmap("Spectral")

ax = total_interest.plot(figsize=(100,20), colormap=cmap,linewidth=5)
ax.legend().remove()
ax.set(title="Interest in Movies (from 01/01/2004 - 02/01/2021)")
ax.set(xlabel='months from 01/01/2004 - 02/01/2021', ylabel='interest')

plt.xticks(list(range(len(movie_interest_df.columns))), movie_interest_df.columns, rota
plt.show()
```

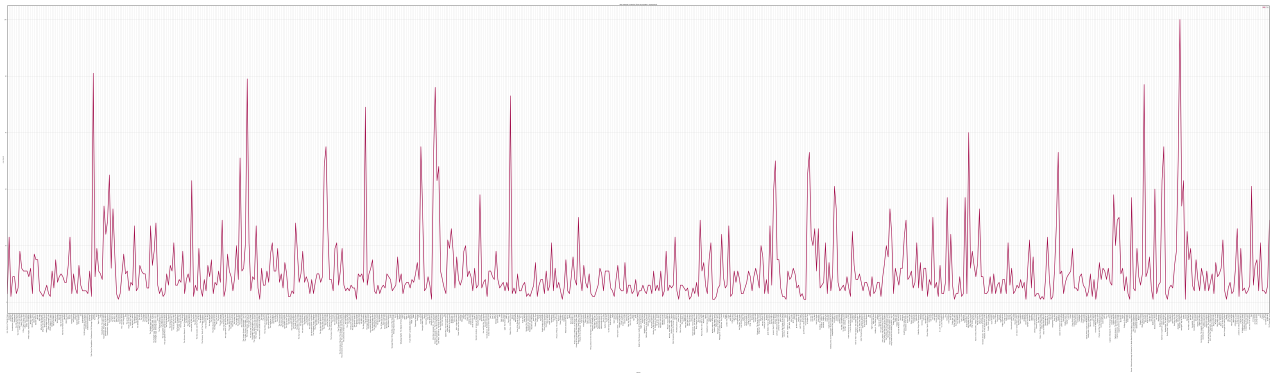


A brief graph to illustrate all the data in the movie_interest dataset. The legend for which lines correlating to what movies was removed on purpose due to the sheer number of movies as we are only interested in this graph out of curiosity for the scope of our data. Curiously, we note that large spikes tend to occur during the summer months. We will explore this further in our linear models.

In [228…

```
# we are interested in the max height of popularity reached by the films
max_popularity = movie_interest_df.transpose().describe().loc['max':]
max_popularity = max_popularity.transpose().reset_index()
max_popularity = max_popularity.sort_values('movie', ascending = False).reset_index(dro

# plot configurations
ax = max_popularity.plot(x='movie', y='max',figsize=(200,50),colormap=cmap,linewidth=5)
ax.set(title="Max Interest in Movies (from 01/01/2004 - 02/01/2021)")
```

```
ax.set(xlabel='movies', ylabel='max interest')
plt.xticks(list(range(len(max_popularity['movie']))), max_popularity['movie'], rotation

ax.grid(b=True, which='major', color='gray', linewidth=0.5)

plt.show()
```



We also analyzed the peak popularity reached by each film. "Peak" is calculated by finding the max popularity for each film over the 206 months of data for movie interest. As expected, the highest spike is Avenger's Endgame, which has a max popularity of 100. Other large spikes of interest include Twilight, Star Wars: The Force Awakens, and Black Panther. Most spikes belong to movies with genres related to action/adventure or romance.

In [229…

```
max_popularity.describe()
```

Out[229…

	max
count	706.000000
mean	10.985836
std	11.937417
min	1.000000
25%	4.000000
50%	7.000000
75%	12.000000
max	100.000000

In [230…

```
# finding quartiles
q1 = max_popularity.describe().iloc[4]
q3 = max_popularity.describe().iloc[6]
iqr = q3-q1
lower_bound = q1 - 1.5*iqr
upper_bound = q3 + 1.5*iqr
lower_bound, upper_bound
```

Out[230…

```
(max   -8.0
 dtype: float64,
 max    24.0
 dtype: float64)
```

In [231…
```
max_popularity.shape
```

Out[231… (706, 2)

We have found that our outliers lie in the range above 24 in terms of search popularity. However, eliminating this data means ridding our dataset of the movies that we are interested in, as we wish to detect the relationship in the success of a movie during its opening weekend release and its search popularity. Therefore, we will keep the data as is.

In [232…
```
movies_below_upper_bound = max_popularity[max_popularity['max'] < 24]
movies_below_upper_bound
```

Out[232…

	movie	max
0	xXx: Return of Xander Cage	3.0
1	Zootopia	23.0
2	Zookeeper	2.0
3	Zombieland: Double Tap	9.0
4	Zombieland	9.0
…	…	…
700	2012	21.0
701	2 Guns	4.0
702	17 Again	4.0
703	13 Going on 30	3.0
704	10,000 BC	6.0

637 rows × 2 columns

Data Analysis & Results (EDA):

Analyzing all opening-weekend earning movies and their search popularity:

In [233…
```
# Loading datasets
no_outliers_opening_weekend_df = pd.read_csv('data/no_outliers.csv')

top_earning_movies = no_outliers_opening_weekend_df[['name', 'opening', 'date']]
top_earning_movies
```

Out[233…

	name	opening	date
0	Iron Man	98618668	2008-05-02
1	Transformers: Dark of the Moon	97852865	2011-06-29

	name	opening	date
2	Fast & Furious 6	97375245	2013-05-24
3	Joker	96202337	2019-10-04
4	Captain America: The Winter Soldier	95023721	2014-04-04
...	...	...	...
699	The Manchurian Candidate	20018620	2004-07-30
700	Pitch Perfect 3	19928525	2017-12-22
701	Ouija	19875995	2014-10-24
702	Kick-Ass	19828687	2010-04-16
703	The Unborn	19810585	2009-01-09

704 rows × 3 columns

In [234...
```
movie_interest_df = movie_interest_df.reset_index()
top_earning_movies_interest = movie_interest_df[movie_interest_df['movie'].isin(list(to
```

In [235...
```
# preparing dataframes for merging and removing day from the dates
top_earning_movies_interest = top_earning_movies_interest.rename(columns={"movie": "nam
top_earning_movies_interest = top_earning_movies_interest.rename(columns = lambda end:
top_earning_movies_interest
```

Out[235...

	name	2004-01	2004-02	2004-03	2004-04	2004-05	2004-06	2004-07	2004-08	2004-09	...	2020-05	2020-06	2
0	Vantage Point	0	0	0	0	0	0	0	0	0	...	0	0	
1	Cars	0	0	0	0	0	0	0	0	0	...	1	1	
2	Big Hero 6	0	0	0	0	0	0	0	0	0	...	1	1	
4	xXx: Return of Xander Cage	0	0	0	0	0	0	0	0	0	...	0	0	
5	Miss Peregrine's Home for Peculiar Children	0	0	0	0	0	0	0	0	0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
701	Kingsman: The Golden Circle	0	0	0	0	0	0	0	0	0	...	0	0	
702	The Fault in Our Stars	0	0	0	0	0	0	0	0	0	...	0	0	

	name	2004-01	2004-02	2004-03	2004-04	2004-05	2004-06	2004-07	2004-08	2004-09	...	2020-05	2020-06	2
703	The Lego Movie	0	0	0	0	0	0	0	0	0	...	1	0	
704	Peter Rabbit	0	0	0	0	0	0	0	0	0	...	0	0	
705	Dawn of the Planet of the Apes	0	0	0	0	0	0	0	0	0	...	0	0	

615 rows × 207 columns

In [236…

```
# merging dataframes: top opening weekend movies and movie interest (merged by name of
top_earning_movies_interest = top_earning_movies_interest.merge(top_earning_movies, how
top_earning_movies_interest
```

Out[236…

	name	2004-01	2004-02	2004-03	2004-04	2004-05	2004-06	2004-07	2004-08	2004-09	...	2020-07	2020-08	2
0	Vantage Point	0	0	0	0	0	0	0	0	0	...	0	0	
1	Cars	0	0	0	0	0	0	0	0	0	...	1	1	
2	Big Hero 6	0	0	0	0	0	0	0	0	0	...	0	0	
3	xXx: Return of Xander Cage	0	0	0	0	0	0	0	0	0	...	0	0	
4	Miss Peregrine's Home for Peculiar Children	0	0	0	0	0	0	0	0	0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
610	Kingsman: The Golden Circle	0	0	0	0	0	0	0	0	0	...	0	0	
611	The Fault in Our Stars	0	0	0	0	0	0	0	0	0	...	0	0	
612	The Lego Movie	0	0	0	0	0	0	0	0	0	...	0	0	
613	Peter Rabbit	0	0	0	0	0	0	0	0	0	...	0	0	

	name	2004-01	2004-02	2004-03	2004-04	2004-05	2004-06	2004-07	2004-08	2004-09	...	2020-07	2020-08	2
614	Dawn of the Planet of the Apes	0	0	0	0	0	0	0	0	0	...	0	0	

615 rows × 209 columns

In [237…

```
# search popularity for all movies in the month prior to their release: these values ar
top_earning_movies_interest['pre_interest'] = range(len(top_earning_movies_interest))
for i, movie in enumerate(list(top_earning_movies_interest['name'])):
    release_date = top_earning_movies_interest[top_earning_movies_interest['name'] == m
    popularity = top_earning_movies_interest.at[i, release_date]
    top_earning_movies_interest.at[i, 'pre_interest'] = popularity
top_earning_movies_interest = top_earning_movies_interest[['name', 'opening', 'pre_inte
top_earning_movies_interest
```

Out[237…

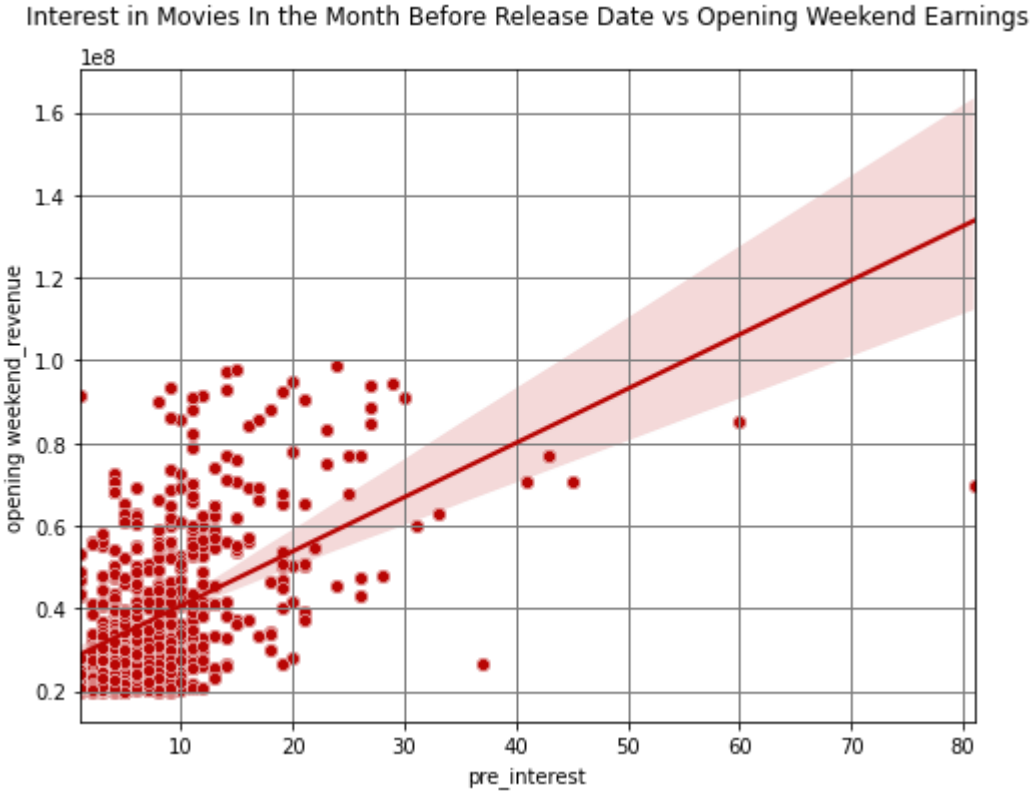
	name	opening	pre_interest
0	Vantage Point	22874936	3
1	Cars	60119509	12
2	Big Hero 6	56215889	16
3	xXx: Return of Xander Cage	20130142	3
4	Miss Peregrine's Home for Peculiar Children	28871140	3
...	...	...	...
610	Kingsman: The Golden Circle	39023010	5
611	The Fault in Our Stars	48002523	28
612	The Lego Movie	69050279	17
613	Peter Rabbit	25010928	7
614	Dawn of the Planet of the Apes	72611427	10

615 rows × 3 columns

In [238…

```
# comparing the interest in movies in the month before the release date to their openin
top_earning_movies_interest = top_earning_movies_interest[top_earning_movies_interest["

# plot configurations
plt.figure(figsize=(8,6))
sns.set_palette(sns.color_palette("OrRd_r"))

ax = sns.regplot(x="pre_interest", y="opening", data=top_earning_movies_interest)
ax2 = sns.scatterplot(x="pre_interest", y="opening", data=top_earning_movies_interest)
ax.set(title="Interest in Movies In the Month Before Release Date vs Opening Weekend Ea
```

```
ax.title.set_position([.5, 1.05])
ax.grid(b=True, which='major', color='gray', linewidth=1.0)
ax.grid(b=True, which='minor', color='gray', linewidth=0.5)
ax.set_xlabel("pre_interest")
ax.set_ylabel("opening weekend_revenue")

plt.show()
```

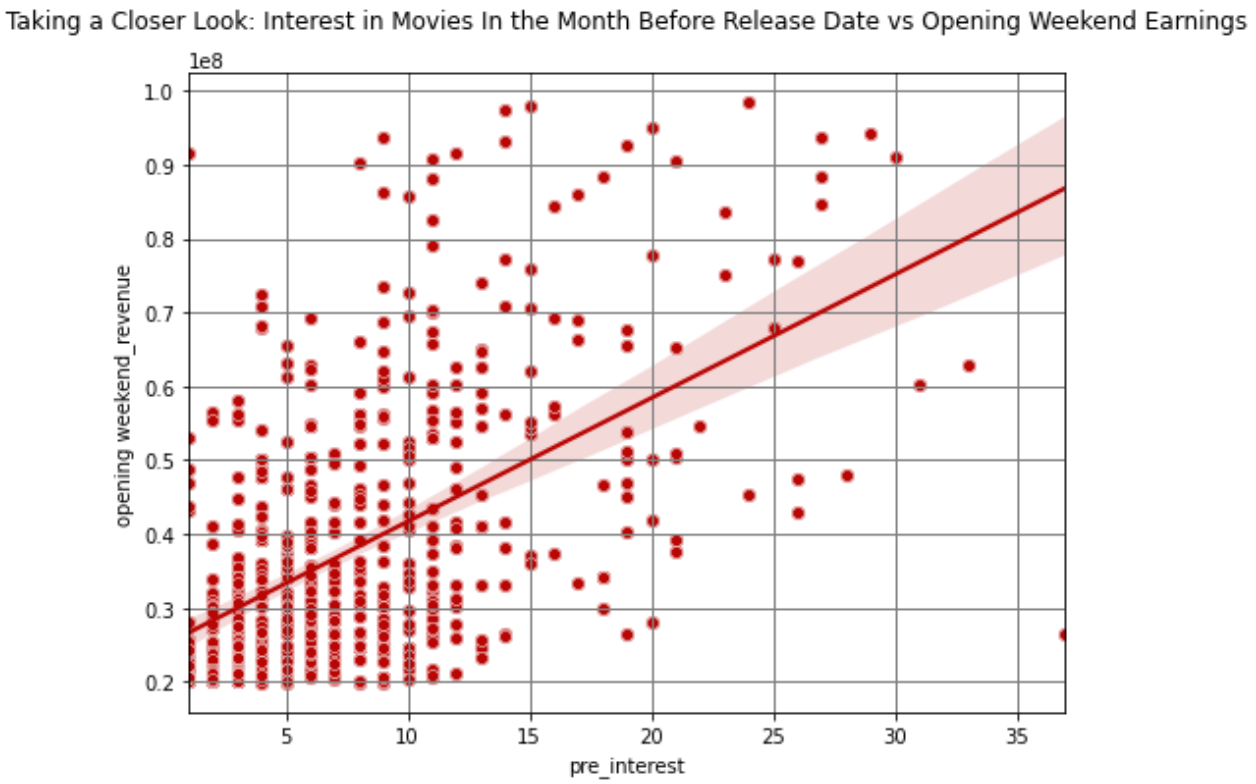


As seen above, the prerelease interest appears to have a positive relationship with opening revenue. There are some outliers, but we left the outliers in for visualization purposes. Now, let's take a look at the graph without the outliers.

In [239…

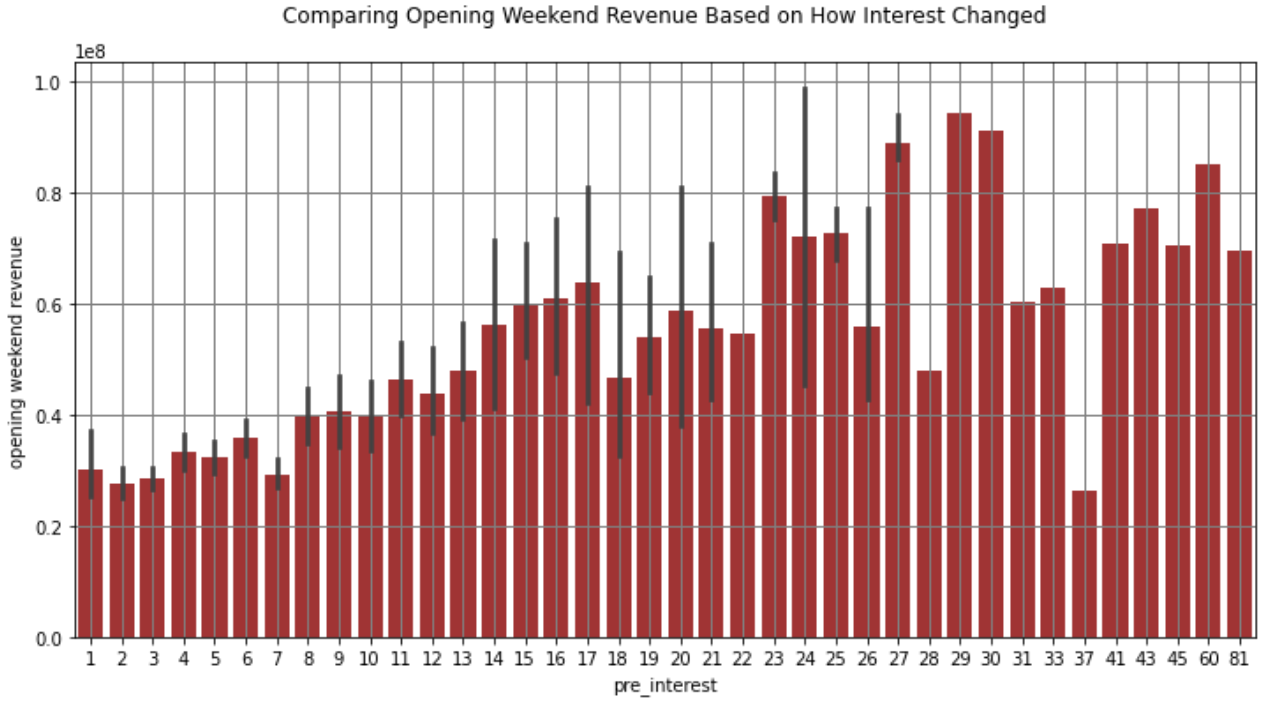
```
# taking a closer look:
# comparing the interest in movies in the month before the release date to their openin

# removing outliers
no_outliers_interest_opening_weekend = top_earning_movies_interest[top_earning_movies_i

# plot configurations
plt.figure(figsize=(8,6))

sns.set_palette(sns.color_palette("OrRd_r"))
ax = sns.regplot(x="pre_interest", y="opening", data=no_outliers_interest_opening_weeke
ax2 = sns.scatterplot(x="pre_interest", y="opening", data=no_outliers_interest_opening_
ax.set(title="Taking a Closer Look: Interest in Movies In the Month Before Release Date

ax.title.set_position([.5, 1.05])
ax.grid(b=True, which='major', color='gray', linewidth=1.0)
ax.grid(b=True, which='minor', color='gray', linewidth=0.5)
ax.set_xlabel("pre_interest")
ax.set_ylabel("opening weekend_revenue")
```

```
plt.show()
```



In [240…

```
# exploring how opening weekend revenue changes based on pre-interest in movies

# plot configurations
plt.figure(figsize=(12,6))

sns.set_palette(sns.color_palette("OrRd_r"), 1)
ax = sns.barplot(x="pre_interest", y="opening", color='firebrick',data=top_earning_movi
ax.set(title="Comparing Opening Weekend Revenue Based on How Interest Changed")

ax.title.set_position([.5, 1.05])
ax.grid(b=True, which='major', color='gray', linewidth=1.0)
ax.grid(b=True, which='minor', color='gray', linewidth=0.5)
ax.set_xlabel("pre_interest")
ax.set_ylabel("opening weekend revenue")

plt.show()
```



In [241…

```
# number of movies with a certain interest
top_earning_movies_interest["pre_interest"].value_counts(sort=False)
```

Out[241…

```
1     27
2     36
3     72
4     64
5     54
6     55
7     39
8     34
9     39
10    29
11    35
12    19
13    14
14    10
15    10
16     5
17     4
18     5
19    10
20     5
21     6
22     1
23     2
24     2
25     2
26     3
27     3
28     1
29     1
30     1
31     1
33     1
37     1
41     1
43     1
```

```
45     1
60     1
81     1
Name: pre_interest, dtype: int64
```

In [242…

```
top_earning_movies_interest[top_earning_movies_interest["pre_interest"] >= 28]
```

Out[242…

	name	opening	pre_interest
10	Avatar	77025481	43
25	Fifty Shades of Grey	85171450	60
27	Transformers	70502384	45
201	It Chapter Two	91062152	30
231	Straight Outta Compton	60200180	31
281	300	70885301	41
339	Borat: Cultural Learnings of America for Make ...	26455463	37
447	Inception	62785337	33
500	Guardians of the Galaxy	94320883	29
557	Twilight	69637740	81
611	The Fault in Our Stars	48002523	28

Note that interest from [28, 81] only consists of a single movie. If we take a closer look at these movies, they all are relatively high-earning movies.

As demonstrated by the scatterplots, movies that have less interest prior to their opening weekend tend to have lower opening weekend revenues. There is a general rise in opening weekend revenue with movies that have interest between 15 and 30. It is interesting to note that the outliers we marked (movies with interest greater than 40) all earned enough by themselves in terms of opening weekend revenue; they far surpass the revenue of numerous movies with varying interests less than 40.

Linear Prediction Models and Variable Relationship Analysis:

First we removed any outliers we previously spotted from the data.

In [243…

```
no_outliers_df = pd.read_csv('data/no_outliers.csv',index_col=0)
movie_interest_df = pd.read_csv('data/movie_interest.csv')
# Remove one movie causing problems with collecting data from month before its release
no_outliers_df = no_outliers_df[no_outliers_df['date'] >= '2004-02-01']

# Format columns of movie interest dataframe to make it easier to work with
new_columns = []
for i in range(len(movie_interest_df.columns)):
    if i == 0:
        new_columns.append("movie")
    else:
        new_column = movie_interest_df.columns[i].split(' ')[0]
        new_columns.append(new_column)
```

```
movie_interest_df.columns = new_columns
movie_interest_df = movie_interest_df.set_index('movie')

# Gather all data without outliers
opening_weekend_df = no_outliers_df[no_outliers_df.name.isin(movie_interest_df.index)]
movie_interest_df = movie_interest_df[movie_interest_df.index.isin(opening_weekend_df.n

opening_weekend_df = opening_weekend_df.reset_index()
assert(opening_weekend_df.shape[0] == movie_interest_df.shape[0])
print("Number of movies: " + str(opening_weekend_df.shape[0]))
opening_weekend_df.head()
```

```
Number of movies: 638
```

Out[243…

	index	rank	name	opening	total	open_percent	theaters	average	date	dis
0	61	62	Iron Man	98618668	318604126	31.0	4105	24024	2008-05-02	Paramour Picture
1	62	63	Transformers: Dark of the Moon	97852865	352390543	27.8	4088	23936	2011-06-29	DreamWork
2	63	64	Fast & Furious 6	97375245	238679850	40.8	3658	26619	2013-05-24	Universa Picture
3	64	65	Joker	96202337	335451311	28.7	4374	21994	2019-10-04	Warner Bro
4	65	66	Captain America: The Winter Soldier	95023721	259766572	36.6	3938	24129	2014-04-04	Walt Disne Studic Motio Picture

In [244…

```
movie_interest_df.head()
```

Out[244…

movie	2004-01-01	2004-02-01	2004-03-01	2004-04-01	2004-05-01	2004-06-01	2004-07-01	2004-08-01	2004-09-01	2004-10-01	…	2020-05-01	2020-06-01
Vantage Point	0	0	0	0	0	0	0	0	0	0	…	0	0
Cars	0	0	0	0	0	0	0	0	0	0	…	1	1
Big Hero 6	0	0	0	0	0	0	0	0	0	0	…	1	1
xXx: Return of Xander Cage	0	0	0	0	0	0	0	0	0	0	…	0	0
Miss Peregrine's Home for Peculiar Children	0	0	0	0	0	0	0	0	0	0	…	0	0

5 rows × 206 columns

Since we could only get the search popularity of a movie on a month-by-month basis, we approximated the number of searches a movie got before release by summing the search popularity of the movie on the month of the release date, and the month before the release date. This captures all the cases where a movie is released in the beginning of the month or at the end of the month. We also separated the dates for ease of use.

In [245…

```
def get_prerelease_search_pop(name, release_date):
    date_arr = release_date.split('-')
    month_of_release = date_arr[0] + '-' +date_arr[1] + "-01"
    year = int(date_arr[0])
    month = int(date_arr[1])
    month -= 1
    if month == 0:
        year -= 1
        if year == 2003:
            print("error, too far in past")
        month = 12
    str_year = str(year)
    str_month = str(month)
    if len(str_month) == 1:
        str_month = '0' + str_month
    month_before_release = str_year + '-' + str_month + "-01"
    pop_sum = movie_interest_df.loc[name,month_of_release] + movie_interest_df.loc[name
    return pop_sum
def get_year(release_date):
    date_arr = release_date.split('-')
    return date_arr[0]
def get_month(release_date):
    date_arr = release_date.split('-')
    return date_arr[1]
def get_day(release_date):
    date_arr = release_date.split('-')
    return date_arr[2]
```

Having written the function to get the approximation of prerelease search popularity, we applied it to the opening weekend data to create a new column called search_pop. We also applied the date separating functions.

In [246…

```
opening_weekend_df['search_pop'] = opening_weekend_df.apply(
    lambda row: get_prerelease_search_pop(row['name'],row['date']),axis=1)
opening_weekend_df['year'] = opening_weekend_df['date'].apply(get_year)
opening_weekend_df['month'] = opening_weekend_df['date'].apply(get_month)
opening_weekend_df['day'] = opening_weekend_df['date'].apply(get_day)

opening_weekend_df.head()
```

Out[246…

	index	rank	name	opening	total	open_percent	theaters	average	date	dis
0	61	62	Iron Man	98618668	318604126	31.0	4105	24024	2008-05-02	Paramour Picture

	index	rank	name	opening	total	open_percent	theaters	average	date	dis
1	62	63	Transformers: Dark of the Moon	97852865	352390543	27.8	4088	23936	2011-06-29	DreamWork
2	63	64	Fast & Furious 6	97375245	238679850	40.8	3658	26619	2013-05-24	Universa Picture
3	64	65	Joker	96202337	335451311	28.7	4374	21994	2019-10-04	Warner Bro
4	65	66	Captain America: The Winter Soldier	95023721	259766572	36.6	3938	24129	2014-04-04	Walt Disne Studic Motio Picture

Once again, we found outliers with search_pop of 0. This indicates that our search was again conflated with some other search term. So we removed these outliers from our data.

In [247…

```
# movies with search_pop 0
opening_weekend_df[opening_weekend_df['search_pop'] == 0]
```

Out[247…

	index	rank	name	opening	total	open_percent	theaters	average	date	d
3	64	65	Joker	96202337	335451311	28.7	4374	21994	2019-10-04	Warner Br
23	88	89	Doctor Strange	85058311	232641920	36.6	3882	21910	2016-11-04	Walt Disr Stud Mot Pictu
28	95	96	Venom	80255756	213515506	37.6	4250	18883	2018-10-05	Sony Pictu Entertainm (S
37	107	108	Fantastic Beasts and Where to Find Them	74403387	234037575	31.8	4144	17954	2016-11-18	Warner Br
42	115	116	Us	71117625	175084580	40.6	3741	19010	2019-03-22	Univer Pictu
58	136	137	Cinderella	67877361	201151353	33.7	3845	17653	2015-03-13	Walt Disr Stud Mot Pictu
63	142	143	Brave	66323594	237283207	28.0	4164	15927	2012-06-22	Walt Disr Stud Mot Pictu
66	145	146	Thor	65723338	181030624	36.3	3955	16617	2011-05-06	Paramou Pictu

	index	rank	name	opening	total	open_percent	theaters	average	date	d
105	196	197	Gravity	55785112	274092705	20.4	3575	15604	2013-10-04	Warner Br
149	255	256	Neighbors	49033915	150157400	32.7	3279	14953	2014-05-09	Univer Pictu
158	266	267	The Longest Yard	47606480	158119460	30.1	3634	13100	2005-05-27	Paramou Pictu
167	280	281	Trolls	46581142	153707064	30.3	4060	11473	2016-11-04	Twenti Century F
187	309	310	The Croods	43639736	187168425	23.3	4046	10785	2013-03-22	Twenti Century F
215	357	358	Cloverfield	40058229	80048433	50.0	3411	11743	2008-01-18	Paramou Pictu
247	401	402	Oblivion	37054485	89107235	41.6	3783	9795	2013-04-19	Univer Pictu
250	407	408	Marley & Me	36357586	143153751	25.4	3480	10447	2008-12-25	Twenti Century F
256	416	417	Robin Hood	36063385	105269730	34.3	3503	10295	2010-05-14	Univer Pictu
267	431	432	Super 8	35451168	127004179	27.9	3379	10491	2011-06-10	Paramou Pictu
270	435	436	Sing	35258145	270395425	13.0	4022	8766	2016-12-21	Univer Pictu
360	572	573	Dark Shadows	29685274	79727149	37.2	3755	7905	2012-05-11	Warner Br
362	574	575	Creed	29632823	109767581	27.0	3404	8705	2015-11-25	Warner Br
366	581	582	Now You See Me	29350389	117723989	24.9	2925	10034	2013-05-31	Lionsg
367	582	583	Beverly Hills Chihuahua	29300465	94514402	31.0	3215	9113	2008-10-03	Walt Disr Stud Mot Pictu
394	623	624	Big Momma's House 2	27736056	70165972	39.5	3261	8505	2006-01-27	Twenti Century F
397	630	631	Wonder	27547866	132422809	20.8	3096	8897	2017-11-17	Lionsg
400	634	635	Beowulf	27515871	82280579	33.4	3153	8726	2007-11-16	Paramou Pictu
433	680	681	The Perfect Guy	25888154	57027435	45.4	2221	11656	2015-09-11	Screen Ge
436	684	685	Rocketman	25725722	96368160	26.7	3610	7126	2019-05-31	Paramou Pictu

	index	rank	name	opening	total	open_percent	theaters	average	date	d
443	695	696	Total Recall	25577758	58877969	43.4	3601	7102	2012-08-03	Sony Pictu Entertainm (S
460	727	728	The Other Woman	24763752	83911193	29.5	3205	7726	2014-04-25	Twenti Century F
464	731	732	Taken	24717037	145000989	17.0	3183	7765	2009-01-30	Twenti Century F
475	746	747	Mean Girls	24432195	86058055	28.4	2839	8605	2004-04-30	Paramou Pictu
481	754	755	No Good Deed	24250283	52543632	46.2	2175	11149	2014-09-12	Screen Ge
487	763	764	White Noise	24113565	56386759	42.8	2261	10665	2005-01-07	Univer Pictu
496	773	774	Bad Moms	23817340	113257297	21.0	3215	7408	2016-07-29	S Entertainm
504	783	784	Tomb Raider	23633317	58250803	40.6	3854	6132	2018-03-16	Warner Br
528	820	821	The Interpreter	22822455	72708161	31.4	2758	8275	2005-04-22	Univer Pictu
551	862	863	Chronicle	22004098	64575175	34.1	2907	7569	2012-02-03	Twenti Century F
592	924	925	Journey to the Center of the Earth	21018141	101704370	20.7	2811	7477	2008-07-11	Warner Br
593	925	926	The Strangers	20997985	52597610	39.9	2466	8514	2008-05-30	Rog Pictu
598	934	935	Prisoners	20817053	61002302	34.1	3260	6385	2013-09-20	Warner Br
611	953	954	Notorious	20497596	36843682	55.6	1638	12513	2009-01-16	F Searchli Pictu
627	981	982	Bewitched	20131130	63313159	31.8	3174	6342	2005-06-24	Sony Pictu Entertainm (S

In [248…

```
# Filter out outliers with 0 search_pop before release date
opening_weekend_df = opening_weekend_df[opening_weekend_df['search_pop'] > 0]
opening_weekend_df.head(opening_weekend_df.shape[0])
opening_weekend_df.to_csv('data/final_data.csv')
```

Having finally gotten all our data together, we used patsy dmatrices and sm.OLS to create a linear predictor, predicting opening revenue based on prerelease search popularity. We will use an alpha

of 0.01 to make inferences. First we make a linear model predicting opening revenue with just search_pop.

In [249…

```
# Predicting opening based on search_pop
outcome_1,predictors_1 = patsy.dmatrices("opening ~ search_pop",opening_weekend_df)
mod_1 = sm.OLS(outcome_1,predictors_1)
res_1 = mod_1.fit()
print(res_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                opening   R-squared:                       0.270
Model:                            OLS   Adj. R-squared:                  0.269
Method:                 Least Squares   F-statistic:                     219.7
Date:                Wed, 17 Mar 2021   Prob (F-statistic):           1.66e-42
Time:                        19:11:22   Log-Likelihood:                -10695.
No. Observations:                 595   AIC:                         2.139e+04
Df Residuals:                     593   BIC:                         2.140e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   2.808e+07    9.4e+05     29.867      0.000    2.62e+07    2.99e+07
search_pop  1.018e+06   6.87e+04     14.821      0.000    8.83e+05    1.15e+06
==============================================================================
Omnibus:                       96.621   Durbin-Watson:                   0.552
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              172.643
Skew:                           0.967   Prob(JB):                     3.24e-38
Kurtosis:                       4.795   Cond. No.                         20.3
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

From the summary, it appears that prerelease search popularity lead to a significant difference in revenue as search_pop has p-value less than 0.01. In addition, there appears to be a positive relationship as the coefficient of search_pop is a positive value. However, we must check for confounding variables to ensure that these results are not just the effect of some other variables. So, next we predict opening revenue with search_pop and the distributor of the movie.

In [250…

```
# Predicting opening based on search_pop and distributor
outcome_1,predictors_1 = patsy.dmatrices("opening ~ search_pop + dist",opening_weekend_
mod_1 = sm.OLS(outcome_1,predictors_1)
res_1 = mod_1.fit()
print(res_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                opening   R-squared:                       0.347
Model:                            OLS   Adj. R-squared:                  0.317
Method:                 Least Squares   F-statistic:                     11.61
Date:                Wed, 17 Mar 2021   Prob (F-statistic):           7.00e-38
Time:                        19:11:23   Log-Likelihood:                -10662.
No. Observations:                 595   AIC:                         2.138e+04
Df Residuals:                     568   BIC:                         2.150e+04
Df Model:                          26
Covariance Type:            nonrobust
```

```
==============================================================================================
=======================
                                                coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------------------------------
----------------------
Intercept                                    1.852e+07    1.5e+07      1.235      0.217
-1.09e+07     4.8e+07
dist[T.CBS Films]                           -7.052e+06   2.12e+07     -0.333      0.740
-4.87e+07    3.46e+07
dist[T.Dimension Films]                     -2.177e+07   2.13e+07     -1.023      0.307
-6.36e+07       2e+07
dist[T.DreamWorks]                           1.686e+07   1.54e+07      1.096      0.274
-1.34e+07    4.71e+07
dist[T.DreamWorks Distribution]              1.015e+07   1.73e+07      0.586      0.558
-2.39e+07    4.42e+07
dist[T.FilmDistrict]                         9.486e+06   1.84e+07      0.517      0.606
-2.66e+07    4.56e+07
dist[T.Focus Features]                      -3.676e+05   1.73e+07     -0.021      0.983
-3.44e+07    3.36e+07
dist[T.Lionsgate]                            2.402e+06   1.52e+07      0.158      0.875
-2.75e+07    3.23e+07
dist[T.Metro-Goldwyn-Mayer (MGM)]            2.947e+06   1.68e+07      0.176      0.861
-3e+07    3.59e+07
dist[T.New Line Cinema]                      1.284e+06   1.62e+07      0.079      0.937
-3.05e+07    3.31e+07
dist[T.Overture Films]                      -6.173e+05   2.12e+07     -0.029      0.977
-4.23e+07     4.1e+07
dist[T.Paramount Pictures]                   1.116e+07   1.51e+07      0.737      0.461
-1.86e+07    4.09e+07
dist[T.Relativity Media]                    -1.202e+06   1.73e+07     -0.069      0.945
-3.52e+07    3.28e+07
dist[T.Revolution Studios]                  -2.031e+06   1.68e+07     -0.121      0.904
-3.5e+07    3.09e+07
dist[T.STX Entertainment]                   -5.338e+06   1.84e+07     -0.291      0.771
-4.14e+07    3.07e+07
dist[T.Screen Gems]                         -1.038e+06   1.55e+07     -0.067      0.946
-3.14e+07    2.93e+07
dist[T.Sony Pictures Entertainment (SPE)]    9.536e+06   1.51e+07      0.632      0.528
-2.01e+07    3.92e+07
dist[T.Summit Entertainment]                -1.182e+07   1.75e+07     -0.675      0.500
-4.62e+07    2.26e+07
dist[T.The Weinstein Company]                5.736e+05   1.73e+07      0.033      0.974
-3.34e+07    3.46e+07
dist[T.TriStar Pictures]                    -3.184e+06   1.62e+07     -0.197      0.844
-3.5e+07    2.86e+07
dist[T.Twentieth Century Fox]                 9.81e+06   1.51e+07      0.650      0.516
-1.98e+07    3.95e+07
dist[T.United Artists]                      -3.765e+06   2.12e+07     -0.178      0.859
-4.54e+07    3.79e+07
dist[T.United Artists Releasing]             7.598e+06   2.12e+07      0.358      0.720
-3.41e+07    4.92e+07
dist[T.Universal Pictures]                   8.261e+06   1.51e+07      0.548      0.584
-2.13e+07    3.79e+07
dist[T.Walt Disney Studios Motion Pictures]  1.702e+07   1.51e+07      1.127      0.260
-1.27e+07    4.67e+07
dist[T.Warner Bros.]                         1.022e+07   1.51e+07      0.678      0.498
-1.94e+07    3.98e+07
search_pop                                   1.045e+06   7.04e+04     14.851      0.000
9.07e+05    1.18e+06
==============================================================================
Omnibus:                       84.002   Durbin-Watson:                   0.698
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              133.635
Skew:                           0.908   Prob(JB):                     9.58e-30
Kurtosis:                       4.446   Cond. No.                     1.72e+03
```

```
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
[2] The condition number is large, 1.72e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

From the summary, we can see that none of the movie distributors are significant, indicating that they are not confounding variables. In addition, search_pop remains significant, with a positive relationship with opening revenue. Next, we predict opening revenue with search_pop and the number of theaters the movie opened in.

In [251…

```
# Predicting opening based on search_pop and theaters
outcome_1,predictors_1 = patsy.dmatrices("opening ~ search_pop + theaters",opening_week
mod_1 = sm.OLS(outcome_1,predictors_1)
res_1 = mod_1.fit()
print(res_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                opening   R-squared:                       0.514
Model:                            OLS   Adj. R-squared:                  0.513
Method:                 Least Squares   F-statistic:                     313.5
Date:                Wed, 17 Mar 2021   Prob (F-statistic):           1.42e-93
Time:                        19:11:23   Log-Likelihood:                -10574.
No. Observations:                 595   AIC:                         2.115e+04
Df Residuals:                     592   BIC:                         2.117e+04
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept  -3.352e+07   3.65e+06     -9.176      0.000   -4.07e+07   -2.63e+07
search_pop  8.698e+05   5.67e+04     15.331      0.000    7.58e+05    9.81e+05
theaters    1.836e+04   1064.450     17.248      0.000    1.63e+04    2.05e+04
==============================================================================
Omnibus:                       64.117   Durbin-Watson:                   1.044
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              112.111
Skew:                           0.685   Prob(JB):                     4.52e-25
Kurtosis:                       4.626   Cond. No.                     2.44e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
[2] The condition number is large, 2.44e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

The number of theaters turns out to be a confounding variable, since it is significant. It also appears to have a positive relationship with revenue. This makes sense because logically, the more theaters a movie opens in, the greater the audience that will be able to watch it. However, since search_pop is still significant, search_pop still appears to be a predictor of search revenue. Next, we predict opening revenue with search_pop and the year of release of the movie.

In [252…

```
# Predicting opening based on search_pop and year
outcome_1,predictors_1 = patsy.dmatrices("opening ~ search_pop + year",opening_weekend_
mod_1 = sm.OLS(outcome_1,predictors_1)
```

```
res_1 = mod_1.fit()
print(res_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                opening   R-squared:                       0.291
Model:                            OLS   Adj. R-squared:                  0.270
Method:                 Least Squares   F-statistic:                     13.94
Date:                Wed, 17 Mar 2021   Prob (F-statistic):           1.79e-33
Time:                        19:11:24   Log-Likelihood:                -10686.
No. Observations:                 595   AIC:                         2.141e+04
Df Residuals:                     577   BIC:                         2.149e+04
Df Model:                          17
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept      2.95e+07   2.64e+06     11.175      0.000    2.43e+07    3.47e+07
year[T.2005] -4.717e+06   3.87e+06     -1.219      0.223   -1.23e+07    2.88e+06
year[T.2006] -6.452e+06   3.65e+06     -1.766      0.078   -1.36e+07    7.24e+05
year[T.2007]  2.582e+05   3.74e+06      0.069      0.945    -7.1e+06    7.61e+06
year[T.2008] -3.537e+06   3.74e+06     -0.945      0.345   -1.09e+07    3.82e+06
year[T.2009] -2.011e+06   3.48e+06     -0.577      0.564   -8.86e+06    4.83e+06
year[T.2010] -1.755e+06   3.52e+06     -0.499      0.618   -8.67e+06    5.16e+06
year[T.2011]  2.472e+06   3.63e+06      0.681      0.496   -4.65e+06     9.6e+06
year[T.2012]  -1.97e+06   3.61e+06     -0.546      0.585   -9.06e+06    5.12e+06
year[T.2013]   9.12e+05    3.5e+06      0.260      0.795   -5.97e+06    7.79e+06
year[T.2014]  3.585e+06   3.53e+06      1.015      0.311   -3.35e+06    1.05e+07
year[T.2015]  1.976e+04   3.78e+06      0.005      0.996    -7.4e+06    7.44e+06
year[T.2016] -4.801e+06   3.63e+06     -1.323      0.186   -1.19e+07    2.33e+06
year[T.2017] -7.873e+05   3.71e+06     -0.212      0.832   -8.08e+06    6.51e+06
year[T.2018] -1.984e+06   3.64e+06     -0.545      0.586   -9.13e+06    5.16e+06
year[T.2019] -1.111e+06   3.69e+06     -0.301      0.763   -8.35e+06    6.13e+06
year[T.2020] -5.913e+06    6.4e+06     -0.924      0.356   -1.85e+07    6.66e+06
search_pop    1.009e+06   6.95e+04     14.509      0.000    8.72e+05    1.15e+06
==============================================================================
Omnibus:                       84.043   Durbin-Watson:                   0.588
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              132.197
Skew:                           0.915   Prob(JB):                     1.97e-29
Kurtosis:                       4.409   Cond. No.                         238.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

None of the years appear to be signficant, while the effect of search_pop remains the same. This was interesting because we believed that the year might affect the search popularity of the movies since computers could have become more ubiquitous as smartphones and other technology emerged. Interestingly, this hypothesis appears to be incorrect, as the summary below shows that year is not a significant predictor or search_pop.

In [253…

```
# Predicting search_pop based on year
outcome_1,predictors_1 = patsy.dmatrices("search_pop ~ year",opening_weekend_df)
mod_1 = sm.OLS(outcome_1,predictors_1)
res_1 = mod_1.fit()
print(res_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             search_pop   R-squared:                       0.025
```

```
Model:                            OLS   Adj. R-squared:                -0.002
Method:                 Least Squares   F-statistic:                    0.9299
Date:                Wed, 17 Mar 2021   Prob (F-statistic):              0.534
Time:                        19:11:24   Log-Likelihood:                -2160.9
No. Observations:                 595   AIC:                             4356.
Df Residuals:                     578   BIC:                             4430.
Df Model:                          16
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept        7.8333      1.546      5.068      0.000       4.797      10.869
year[T.2005]     1.8908      2.314      0.817      0.414      -2.654       6.436
year[T.2006]     0.9444      2.186      0.432      0.666      -3.349       5.238
year[T.2007]     3.6818      2.235      1.647      0.100      -0.708       8.072
year[T.2008]     3.6515      2.235      1.634      0.103      -0.739       8.042
year[T.2009]     1.5303      2.084      0.734      0.463      -2.563       5.624
year[T.2010]     0.3095      2.107      0.147      0.883      -3.828       4.447
year[T.2011]     0.1937      2.171      0.089      0.929      -4.071       4.458
year[T.2012]     2.5351      2.157      1.175      0.240      -1.702       6.772
year[T.2013]     1.6783      2.095      0.801      0.423      -2.437       5.793
year[T.2014]     4.2381      2.107      2.012      0.045       0.101       8.375
year[T.2015]     4.0417      2.253      1.794      0.073      -0.384       8.467
year[T.2016]     1.1937      2.171      0.550      0.583      -3.071       5.458
year[T.2017]     3.4314      2.218      1.547      0.122      -0.925       7.788
year[T.2018]     4.1667      2.171      1.919      0.055      -0.098       8.431
year[T.2019]     3.3952      2.202      1.542      0.124      -0.929       7.719
year[T.2020]     1.3095      3.831      0.342      0.733      -6.215       8.834
==============================================================================
Omnibus:                      528.755   Durbin-Watson:                   1.508
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            19044.901
Skew:                           3.790   Prob(JB):                         0.00
Kurtosis:                      29.660   Cond. No.                         17.9
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

Next we predict the revenue based on search_pop and month of release. Based on the summary, the only the months of May, June, and July are significant, all with positive coefficents, suggesting that perhaps summer is the best season to release a movie. These months are confounding variables, but search_pop remains significant.

In [254…

```
# Predicting opening based on search_pop and month
outcome_1,predictors_1 = patsy.dmatrices("opening ~ search_pop + month",opening_weekend
mod_1 = sm.OLS(outcome_1,predictors_1)
res_1 = mod_1.fit()
print(res_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                opening   R-squared:                       0.373
Model:                            OLS   Adj. R-squared:                  0.360
Method:                 Least Squares   F-statistic:                     28.85
Date:                Wed, 17 Mar 2021   Prob (F-statistic):           1.39e-51
Time:                        19:11:25   Log-Likelihood:                -10650.
No. Observations:                 595   AIC:                         2.133e+04
Df Residuals:                     582   BIC:                         2.138e+04
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
```

```
                  coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     2.22e+07   2.88e+06      7.718      0.000    1.66e+07    2.79e+07
month[T.02]   2.89e+06   3.47e+06      0.833      0.405   -3.92e+06     9.7e+06
month[T.03]  6.866e+06   3.43e+06      2.002      0.046    1.31e+05    1.36e+07
month[T.04]  4.428e+05   3.77e+06      0.118      0.906   -6.95e+06    7.84e+06
month[T.05]  2.118e+07   3.52e+06      6.015      0.000    1.43e+07    2.81e+07
month[T.06]  1.097e+07   3.33e+06      3.297      0.001    4.43e+06    1.75e+07
month[T.07]  9.255e+06   3.34e+06      2.773      0.006     2.7e+06    1.58e+07
month[T.08]  1.079e+06   3.59e+06      0.301      0.764   -5.97e+06    8.12e+06
month[T.09]  9.809e+05   3.62e+06      0.271      0.787   -6.13e+06    8.09e+06
month[T.10]  8.935e+04   3.63e+06      0.025      0.980   -7.03e+06    7.21e+06
month[T.11]  6.623e+06   3.48e+06      1.904      0.057    -2.1e+05    1.35e+07
month[T.12]   5.82e+06   3.62e+06      1.607      0.109    -1.3e+06    1.29e+07
search_pop   9.669e+05   6.54e+04     14.785      0.000    8.38e+05     1.1e+06
==============================================================================
Omnibus:                       66.478   Durbin-Watson:                   0.730
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              109.783
Skew:                           0.728   Prob(JB):                     1.45e-24
Kurtosis:                       4.519   Cond. No.                         232.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

We ran the same test on predicting opening revenue with search_pop and day of release, but none of the days turned out to be signficant.

In [255…

```
# Predicting opening based on search_pop and day
outcome_1,predictors_1 = patsy.dmatrices("opening ~ search_pop + day",opening_weekend_d
mod_1 = sm.OLS(outcome_1,predictors_1)
res_1 = mod_1.fit()
print(res_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                opening   R-squared:                       0.328
Model:                            OLS   Adj. R-squared:                  0.291
Method:                 Least Squares   F-statistic:                     8.868
Date:                Wed, 17 Mar 2021   Prob (F-statistic):           3.81e-32
Time:                        19:11:25   Log-Likelihood:                -10670.
No. Observations:                 595   AIC:                         2.140e+04
Df Residuals:                     563   BIC:                         2.154e+04
Df Model:                          31
Covariance Type:            nonrobust
==============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   2.247e+07    3.8e+06      5.914      0.000     1.5e+07    2.99e+07
day[T.02]   1.092e+07   4.99e+06      2.190      0.029    1.12e+06    2.07e+07
day[T.03]   2.513e+06   5.36e+06      0.469      0.640   -8.02e+06     1.3e+07
day[T.04]   8.985e+06   5.63e+06      1.596      0.111   -2.07e+06       2e+07
day[T.05]   4.517e+05   5.24e+06      0.086      0.931   -9.85e+06    1.08e+07
day[T.06]   5.256e+06   5.04e+06      1.043      0.298   -4.64e+06    1.52e+07
day[T.07]    3.73e+06    5.1e+06      0.731      0.465   -6.29e+06    1.37e+07
day[T.08]   1.326e+07   5.52e+06      2.403      0.017    2.42e+06    2.41e+07
day[T.09]   2.114e+06   4.76e+06      0.444      0.657   -7.25e+06    1.15e+07
day[T.10]   2.507e+06   4.67e+06      0.537      0.591   -6.66e+06    1.17e+07
day[T.11]   1.701e+06   4.68e+06      0.364      0.716   -7.49e+06    1.09e+07
day[T.12]  -1.337e+06   5.33e+06     -0.251      0.802   -1.18e+07    9.13e+06
day[T.13]   3.321e+06   4.67e+06      0.712      0.477   -5.84e+06    1.25e+07
day[T.14]   3.218e+06   4.93e+06      0.652      0.514   -6.47e+06    1.29e+07
```

```
day[T.15]   5.727e+06    5.1e+06      1.123      0.262   -4.29e+06    1.57e+07
day[T.16]   4.805e+06    4.8e+06      1.000      0.318   -4.63e+06    1.42e+07
day[T.17]   5.269e+06   4.93e+06      1.068      0.286   -4.42e+06     1.5e+07
day[T.18]  -2.939e+06   5.17e+06     -0.569      0.570   -1.31e+07    7.21e+06
day[T.19]   2.963e+06   5.32e+06      0.557      0.578   -7.49e+06    1.34e+07
day[T.20]   4.582e+06   5.25e+06      0.873      0.383   -5.72e+06    1.49e+07
day[T.21]   3.397e+06    4.7e+06      0.723      0.470   -5.83e+06    1.26e+07
day[T.22]   4.471e+06    4.9e+06      0.913      0.362   -5.15e+06    1.41e+07
day[T.23]   7.939e+06   5.43e+06      1.463      0.144   -2.72e+06    1.86e+07
day[T.24]   1.032e+07   5.25e+06      1.966      0.050    9919.904    2.06e+07
day[T.25]    7.27e+06   5.11e+06      1.423      0.155   -2.77e+06    1.73e+07
day[T.26]   8.727e+06   5.25e+06      1.661      0.097   -1.59e+06     1.9e+07
day[T.27]   7.621e+06   5.52e+06      1.380      0.168   -3.23e+06    1.85e+07
day[T.28]    4.28e+06   4.96e+06      0.863      0.388   -5.46e+06     1.4e+07
day[T.29]   1.826e+07   5.33e+06      3.429      0.001     7.8e+06    2.87e+07
day[T.30]   1.311e+07   5.64e+06      2.325      0.020    2.03e+06    2.42e+07
day[T.31]   1.855e+07   7.27e+06      2.550      0.011    4.26e+06    3.28e+07
search_pop  1.067e+06   7.23e+04     14.745      0.000    9.24e+05    1.21e+06
==============================================================================
Omnibus:                       72.296   Durbin-Watson:                   0.666
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              128.120
Skew:                           0.753   Prob(JB):                     1.51e-28
Kurtosis:                       4.702   Cond. No.                         459.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

When we predicted opening revenue with release month, and number of opening theaters we find that only May is the remaining significant month. This suggests that the higher summer opening revenue is actually mostly explained by the number of opening theaters in summer. This means that the increase summer revenue could be caused mainly by the movie industry choosing to use more theaters in summer, rather than summer inherently causing more people to see movies. This is supported by the fact that summer months are significant predictors of number of opening theaters even when predicting number of opening theaters based on distributor and release month as shown below.

In [256…

```
# Predicting opening based on month and theaters
outcome_1,predictors_1 = patsy.dmatrices("opening ~ month+theaters",opening_weekend_df)
mod_1 = sm.OLS(outcome_1,predictors_1)
res_1 = mod_1.fit()
print(res_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                opening   R-squared:                       0.368
Model:                            OLS   Adj. R-squared:                  0.355
Method:                 Least Squares   F-statistic:                     28.19
Date:                Wed, 17 Mar 2021   Prob (F-statistic):           1.58e-50
Time:                        19:11:26   Log-Likelihood:                -10652.
No. Observations:                 595   AIC:                         2.133e+04
Df Residuals:                     582   BIC:                         2.139e+04
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   -2.904e+07   4.87e+06     -5.962      0.000   -3.86e+07   -1.95e+07
month[T.02]  9.586e+05   3.49e+06      0.275      0.784    -5.9e+06    7.82e+06
```

```
month[T.03]  2.365e+06   3.47e+06      0.682      0.496   -4.45e+06    9.18e+06
month[T.04]  5.648e+05   3.78e+06      0.149      0.881   -6.86e+06    7.99e+06
month[T.05]  1.204e+07   3.64e+06      3.305      0.001    4.88e+06    1.92e+07
month[T.06]    6.2e+06   3.39e+06      1.831      0.068   -4.52e+05    1.29e+07
month[T.07]  5.544e+06   3.39e+06      1.636      0.102   -1.11e+06    1.22e+07
month[T.08]  1.366e+06    3.6e+06      0.379      0.705   -5.71e+06    8.44e+06
month[T.09] -4.741e+06   3.66e+06     -1.294      0.196   -1.19e+07    2.45e+06
month[T.10] -1.103e+06   3.65e+06     -0.302      0.762   -8.27e+06    6.06e+06
month[T.11]  5.631e+06    3.5e+06      1.607      0.109   -1.25e+06    1.25e+07
month[T.12]  3.382e+06   3.66e+06      0.924      0.356    -3.8e+06    1.06e+07
theaters     1.866e+04   1282.471     14.551      0.000    1.61e+04    2.12e+04
==============================================================================
Omnibus:                       75.336   Durbin-Watson:                   0.724
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              106.311
Skew:                           0.896   Prob(JB):                     8.22e-24
Kurtosis:                       4.037   Cond. No.                     5.87e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
[2] The condition number is large, 5.87e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Below, we can see in the summary that May, June, and July are all significant when predicting number of opening weekends based on release month and distributor. The fact that none of the distributors are significant suggests that they all use the same strategy when it comes to selecting number of opening theaters based on month.

In [257…

```
# Predicting theaters based on release month and distributor
outcome_1,predictors_1 = patsy.dmatrices("theaters ~ month+dist",opening_weekend_df)
mod_1 = sm.OLS(outcome_1,predictors_1)
res_1 = mod_1.fit()
print(res_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               theaters   R-squared:                       0.268
Model:                            OLS   Adj. R-squared:                  0.220
Method:                 Least Squares   F-statistic:                     5.663
Date:                Wed, 17 Mar 2021   Prob (F-statistic):           7.37e-21
Time:                        19:11:26   Log-Likelihood:                -4441.5
No. Observations:                 595   AIC:                             8957.
Df Residuals:                     558   BIC:                             9119.
Df Model:                          36
Covariance Type:            nonrobust
==============================================================================
                                                 coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
Intercept                                    3584.4796    448.994      7.983      0.000
2702.554    4466.406
month[T.02]                                   167.5204    106.821      1.568      0.117
-42.300     377.340
month[T.03]                                   218.7660    106.459      2.055      0.040
9.657     427.875
month[T.04]                                   160.1959    116.690      1.373      0.170
-69.009     389.401
month[T.05]                                   530.2439    108.603      4.882      0.000
316.923     743.565
```

```
month[T.06]                                    349.7779    101.911      3.432      0.001
149.601     549.955
month[T.07]                                    337.1055    102.363      3.293      0.001
136.041     538.170
month[T.08]                                    164.8321    110.642      1.490      0.137
-52.494     382.158
month[T.09]                                    304.3046    110.604      2.751      0.006
87.054     521.556
month[T.10]                                     98.1177    112.372      0.873      0.383
-122.605     318.841
month[T.11]                                    188.5166    107.459      1.754      0.080
-22.557     399.590
month[T.12]                                    272.6859    111.379      2.448      0.015
53.913     491.459
dist[T.CBS Films]                             -897.0000    616.742     -1.454      0.146
-2108.420     314.420
dist[T.Dimension Films]                       -514.6756    624.567     -0.824      0.410
-1741.465     712.114
dist[T.DreamWorks]                            -101.6294    451.772     -0.225      0.822
-989.012     785.753
dist[T.DreamWorks Distribution]               -322.8094    508.596     -0.635      0.526
-1321.806     676.187
dist[T.FilmDistrict]                          -772.5149    539.593     -1.432      0.153
-1832.397     287.367
dist[T.Focus Features]                        -685.0958    508.556     -1.347      0.178
-1684.014     313.823
dist[T.Lionsgate]                             -820.4082    446.954     -1.836      0.067
-1698.326      57.510
dist[T.Metro-Goldwyn-Mayer (MGM)]             -595.4793    490.624     -1.214      0.225
-1559.175     368.216
dist[T.New Line Cinema]                       -610.0619    474.575     -1.285      0.199
-1542.234     322.110
dist[T.Overture Films]                        -792.5973    623.798     -1.271      0.204
-2017.876     432.682
dist[T.Paramount Pictures]                    -354.0940    444.627     -0.796      0.426
-1227.441     519.253
dist[T.Relativity Media]                      -634.3321    504.391     -1.258      0.209
-1625.069     356.405
dist[T.Revolution Studios]                    -799.1801    493.438     -1.620      0.106
-1768.404     170.044
dist[T.STX Entertainment]                     -571.6319    540.566     -1.057      0.291
-1633.425     490.161
dist[T.Screen Gems]                           -952.4584    452.897     -2.103      0.036
-1842.049     -62.868
dist[T.Sony Pictures Entertainment (SPE)]     -362.4833    442.592     -0.819      0.413
-1231.833     506.867
dist[T.Summit Entertainment]                   -97.1665    508.487     -0.191      0.849
-1095.950     901.617
dist[T.The Weinstein Company]                 -749.2630    509.833     -1.470      0.142
-1750.690     252.164
dist[T.TriStar Pictures]                      -803.0028    476.518     -1.685      0.093
-1738.992     132.986
dist[T.Twentieth Century Fox]                 -323.2443    443.098     -0.730      0.466
-1193.588     547.100
dist[T.United Artists]                       -1146.1656    623.632     -1.838      0.067
-2371.120      78.788
dist[T.United Artists Releasing]               324.4027    623.798      0.520      0.603
-900.876    1549.682
dist[T.Universal Pictures]                    -546.2791    442.237     -1.235      0.217
-1414.931     322.373
dist[T.Walt Disney Studios Motion Pictures]   -182.5025    443.752     -0.411      0.681
-1054.131     689.126
dist[T.Warner Bros.]                          -275.4955    442.089     -0.623      0.533
-1143.857     592.866
==============================================================================
```

```
Omnibus:                       52.555   Durbin-Watson:                   1.603
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              130.291
Skew:                          -0.464   Prob(JB):                     5.10e-29
Kurtosis:                       5.097   Cond. No.                         139.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

In this summary, we can see that month has a weak correlation with theaters, since the r-squared value is low when predicting number of theaters just on release month.

In [258…

```
# Predicting theaters based on just release month
outcome_1,predictors_1 = patsy.dmatrices("theaters ~ month",opening_weekend_df)
mod_1 = sm.OLS(outcome_1,predictors_1)
res_1 = mod_1.fit()
print(res_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               theaters   R-squared:                       0.108
Model:                            OLS   Adj. R-squared:                  0.092
Method:                 Least Squares   F-statistic:                     6.443
Date:                Wed, 17 Mar 2021   Prob (F-statistic):           3.80e-10
Time:                        19:11:26   Log-Likelihood:                -4500.0
No. Observations:                 595   AIC:                             9024.
Df Residuals:                     583   BIC:                             9077.
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    3074.5385     92.320     33.303      0.000    2893.218    3255.859
month[T.02]   224.6838    112.369      2.000      0.046       3.987     445.380
month[T.03]   358.0995    111.102      3.223      0.001     139.890     576.309
month[T.04]   191.6615    121.879      1.573      0.116     -47.713     431.036
month[T.05]   716.3815    113.820      6.294      0.000     492.834     939.929
month[T.06]   486.1054    107.511      4.521      0.000     274.949     697.261
month[T.07]   462.3088    107.707      4.292      0.000     250.768     673.850
month[T.08]   213.0838    115.963      1.838      0.067     -14.673     440.840
month[T.09]   339.2711    117.470      2.888      0.004     108.555     569.987
month[T.10]   207.5568    117.470      1.767      0.078     -23.159     438.273
month[T.11]   382.8797    112.036      3.417      0.001     162.836     602.923
month[T.12]   407.9267    116.946      3.488      0.001     178.239     637.614
==============================================================================
Omnibus:                       39.386   Durbin-Watson:                   1.570
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               66.295
Skew:                          -0.464   Prob(JB):                     4.02e-15
Kurtosis:                       4.346   Cond. No.                         17.6
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

So with respect to predicting opening revenue, the significant predictors were prerelease search popularity, whether or not a movie was released in May, and number of theaters showing the movie for opening. All of these predictors showed a positive relationship with opening revenue. While some confounding was caused by the other variables than prerelease search popularity, the effect of

these variables was not enough to make the effect of prerelease search popularity insignificant. To examine correlation strength, we try predicting opening revenue based on each individual variable.

In [259…

```
# Predicting opening revenue based on just release month
outcome_1,predictors_1 = patsy.dmatrices("opening ~ month",opening_weekend_df)
mod_1 = sm.OLS(outcome_1,predictors_1)
res_1 = mod_1.fit()
print(res_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                opening   R-squared:                       0.137
Model:                            OLS   Adj. R-squared:                  0.121
Method:                 Least Squares   F-statistic:                     8.447
Date:                Wed, 17 Mar 2021   Prob (F-statistic):           6.82e-14
Time:                        19:11:26   Log-Likelihood:                -10745.
No. Observations:                 595   AIC:                         2.151e+04
Df Residuals:                     583   BIC:                         2.157e+04
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    2.834e+07   3.34e+06      8.495      0.000    2.18e+07    3.49e+07
month[T.02]  5.152e+06   4.06e+06      1.269      0.205   -2.82e+06    1.31e+07
month[T.03]  9.048e+06   4.01e+06      2.254      0.025    1.16e+06    1.69e+07
month[T.04]  4.142e+06    4.4e+06      0.940      0.347   -4.51e+06    1.28e+07
month[T.05]  2.541e+07   4.11e+06      6.178      0.000    1.73e+07    3.35e+07
month[T.06]  1.527e+07   3.88e+06      3.931      0.000    7.64e+06    2.29e+07
month[T.07]  1.417e+07   3.89e+06      3.641      0.000    6.53e+06    2.18e+07
month[T.08]  5.343e+06   4.19e+06      1.275      0.203   -2.89e+06    1.36e+07
month[T.09]   1.59e+06   4.24e+06      0.375      0.708   -6.75e+06    9.93e+06
month[T.10]   2.77e+06   4.24e+06      0.653      0.514   -5.57e+06    1.11e+07
month[T.11]  1.278e+07   4.05e+06      3.156      0.002    4.83e+06    2.07e+07
month[T.12]  1.099e+07   4.23e+06      2.602      0.010     2.7e+06    1.93e+07
==============================================================================
Omnibus:                      106.844   Durbin-Watson:                   0.282
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              169.615
Skew:                           1.146   Prob(JB):                     1.47e-37
Kurtosis:                       4.262   Cond. No.                         17.6
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

In [260…

```
# Predicting opening revenue based on just number of opening theaters
outcome_1,predictors_1 = patsy.dmatrices("opening ~ theaters",opening_weekend_df)
mod_1 = sm.OLS(outcome_1,predictors_1)
res_1 = mod_1.fit()
print(res_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                opening   R-squared:                       0.322
Model:                            OLS   Adj. R-squared:                  0.320
Method:                 Least Squares   F-statistic:                     281.0
Date:                Wed, 17 Mar 2021   Prob (F-statistic):           6.43e-52
Time:                        19:11:26   Log-Likelihood:                -10673.
No. Observations:                 595   AIC:                         2.135e+04
Df Residuals:                     593   BIC:                         2.136e+04
```

```
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept  -3.324e+07   4.31e+06     -7.706      0.000   -4.17e+07   -2.48e+07
theaters    2.083e+04   1242.574     16.764      0.000    1.84e+04    2.33e+04
==============================================================================
Omnibus:                       82.402   Durbin-Watson:                   0.645
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              119.340
Skew:                           0.954   Prob(JB):                     1.22e-26
Kurtosis:                       4.085   Cond. No.                     2.44e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
[2] The condition number is large, 2.44e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [261…

```
# Predicting opening revenue based on just prerelease search popularity
outcome_1,predictors_1 = patsy.dmatrices("opening ~ search_pop",opening_weekend_df)
mod_1 = sm.OLS(outcome_1,predictors_1)
res_1 = mod_1.fit()
print(res_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                opening   R-squared:                       0.270
Model:                            OLS   Adj. R-squared:                  0.269
Method:                 Least Squares   F-statistic:                     219.7
Date:                Wed, 17 Mar 2021   Prob (F-statistic):           1.66e-42
Time:                        19:11:26   Log-Likelihood:                -10695.
No. Observations:                 595   AIC:                         2.139e+04
Df Residuals:                     593   BIC:                         2.140e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   2.808e+07    9.4e+05     29.867      0.000    2.62e+07    2.99e+07
search_pop  1.018e+06   6.87e+04     14.821      0.000    8.83e+05    1.15e+06
==============================================================================
Omnibus:                       96.621   Durbin-Watson:                   0.552
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              172.643
Skew:                           0.967   Prob(JB):                     3.24e-38
Kurtosis:                       4.795   Cond. No.                         20.3
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specifi
ed.
```

From the above summaries, since a higher R-squared value means more correlation, the number of theaters was higher correlated than search_pop, which was higher correlated than month. Still, all these variables had low R-squared variables, indicating a medium to weak positive association for all these variables to opening revenue.

Summary of Relationships:

We found that whether or not a movie was released in May had a weak positive relationship with opening revenue. We also found that the number of opening theaters, and prerelease search popularity had moderate positive relationships with opening revenue. Other than predicting opening revenue, we found that the summer months had a positive relationship with the number of opening theaters, regardless of distributor. We also failed to find a relationship between prerelease search popularity and the year a movie was released.

With respect to our research question, we found that prerelease search popularity had a significant positive relationship with opening revenue, regardless of the other variables used to predict opening revenue with it.

Ethics & Privacy

In this project, we used publicly available data to explore the relationship between Google searches and the revenue movies will make. Thus, the stakeholders in our project are Google and the movie industry. Our project will affect them by potentially showing whether or not Google searches correlate to movie revenue. While this may affect the strategies that Google and the movie industry use to promote movies, this will not negatively impact the movie industry or Google; rather, it will simply comment on how these two stakeholders are related. Since our project will not make any judgements that harm either party, it is unlikely that the information we gain from our project would be co-opted for nefarious purposes. In fact, the only possible potential unintended consequence we foresee is that the movie industry changes their strategy of marketing for the better based on what we discover.

Since a film's box office sales and Google searches pertaining to certain films are the data at hand, the level of risk of our data extends to the anonymization of a person's Google searches. We mitigated the risks of our data by avoiding data sets that infringe the privacy of humans' internet searches. All subjects are equally vulnerable, as every human is equally at risk of having their data about their Google searches used without their knowledge. With respect to informed consent, since we will not be collecting data from individuals ourselves, we do not need to obtain informed consent from anyone. The data that we found and analyzed was not tied to any person's Google searches. We tracked different box-office revenues and Google search results, which are dependent on the existence of a high-grossing film with Google searches available for the days of the film's opening weekend. Throughout our analysis, the only instance of bias we found is that our opening weekend revenue data is skewed towards movies with higher revenue, since this data is taken from the top 1000 opening weekend revenues. This means our conclusions may only apply to higher earning movies, rather than lower earning movies, which has the potential to help the big players in the movie industry more than the indie/small companies in the movie industry. However, we believe this difference in aid will not be drastic because our results are not extremely impactful. Other than this, we did not encounter any blaring biases in our data and tried to explore all possible variables, including confounding variables, that could potentially correlate to or falsely predict movies' opening revenues.

In considering the privacy of the data we will analyze, we recognize that our Github repository will be set to public after the completion of this project, so we plan to guarantee secure data storage by implementing file access permissions. No further measures need to be taken, seeing as the amounts of Google searches for a keyword are not tied to specific persons; there is no way for a potential breach to leak private information.

Conclusion & Discussion

We wanted to find out whether or not there is a relationship between the number of Google searches a movie receives before its release and the box-office revenue it makes during its opening weekend. We used datasets that include information about opening weekend dates, box office revenues, and Google search popularity of movies prior to their opening weekends to answer our question. A limitation of our project was that we weren't able to analyze all 1000 movies from the Top Opening Weekends dataset in conjunction with Google's search data because Google Trends doesn't cover the timeframe before January 1st, 2004. As a result, the movies that were released between January 1st, 1997 and January 1st, 2004 were excluded in our second dataset.

After performing our data analysis, we found that Warner Bros. and Universal Pictures have accumulated the highest total opening weekend revenue in comparison to other movie distributors. However, no matter how big or well-known a distributor is, if a distributor chooses an effective time, such as the summer season, to release their movie in a myriad of theaters, their revenue is likely to increase relative to other months. In line with this finding, we discovered through our linear model that an increase in theaters showing the movie in its opening weekend along with an increase in prerelease search popularity have a positive impact on garnering opening weekend revenue. We also found that a movie's search popularity in the month before their opening weekend generally correlates to their opening weekend revenue. This shows us that pushing marketing tactics a month before release could aid in increasing opening weekend turnout. In regards to our research question, prerelease Google search popularity has a significant positive relationship with opening weekend revenue, regardless of the other variables used to predict opening revenue with it. Subsequently, we found evidence that our hypothesis was correct.

If our project is seen by the movie industry, our work could impact the strategies in which they market and distribute their movies. Note that our data consists of the top box-office earnings movies, so our model is inevitably biased towards movies that are more profitable. In this way, our analyses could benefit movie distributors who have a small total opening weekend revenue in comparison to more well-known distributors. If smaller distributors choose to release more of their movies in the summer, they could potentially generate a larger profit. This could lead to a prolongation of their business to produce more movies in the long run and create more content for viewers to see.

Team Contributions

- Megan Tan: conceptual planning of research question/hypothesis, background/prior work, ethics and privacy, data cleaning writeup, some EDA explanations/steps, conclusion/discussion,

presentation slidedeck

- Nhu Luong: conceptual planning of research question/hypothesis, background/prior work, ethics and privacy, data cleaning writeup, presentation slidedeck, presentation
- Ethan Zhou: overview, ethics and privacy, removed outliers from dataframe in EDA, presentation slidedeck, presentation script, presentation
- David Nguyen: conceptual planning of research question/hypothesis, hypothesis justification writeup, ethics and privacy, found dataset, data cleaning + compiled movie revenue data set, linear model + explanations, presentation
- Jennifer Yang: conceptual planning of research question/hypothesis, ethics and privacy, data cleaning + removing outliers for both datasets, data analysis visualizations + explanations, presentation

In [ ]: