{

"cells": [

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"# Movie Industry Exploratory Data Analysis"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"## Objective: Investigate the film industry to gain sufficient understanding of what attributes to success and in turn utilize this analysis to create \*actionable\* recommendations for companies to enter the industry."

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Importing necessary libraries and the datasets."

]

},

{

"cell\_type": "code",

"execution\_count": 1,

"metadata": {

"ExecuteTime": {

"end\_time": "2020-07-17T19:44:34.418023Z",

"start\_time": "2020-07-17T19:44:32.213764Z"

}

},

"outputs": [],

"source": [

"import pandas as pd\n",

"import seaborn as sns\n",

"import matplotlib.pyplot as plt\n",

"import datetime as dt\n",

"import numpy as np"

]

},

{

"cell\_type": "code",

"execution\_count": 57,

"metadata": {},

"outputs": [],

"source": [

"#Setting the default style for plots\n",

"plt.style.use('ggplot')\n",

"\n",

"from matplotlib.pyplot import figure\n",

"plt.rcParams['figure.figsize'] = (12,8)\n",

"\n",

"%matplotlib inline"

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"metadata": {

"ExecuteTime": {

"end\_time": "2020-07-17T19:45:00.483664Z",

"start\_time": "2020-07-17T19:45:00.320561Z"

}

},

"outputs": [],

"source": [

"movie\_dates\_df = pd.read\_csv('movie\_release\_dates.csv', index\_col=0)\n",

"theaters\_df = pd.read\_csv('movie\_theater\_data.csv', index\_col=0)\n",

"awards\_df = pd.read\_csv('movie\_awards.csv', index\_col=0)\n",

"actors\_df = pd.read\_csv('Actors\_Table.csv')\n",

"directors\_df = pd.read\_csv('Directors\_Table.csv')\n",

"imdb\_base\_df = pd.read\_csv('IMDb\_base.csv')\n",

"imdb\_budgets\_df = pd.read\_csv('IMDb\_budgets.csv')\n",

"studio\_df = pd.read\_csv('studiodf.csv')"

]

},

{

"cell\_type": "code",

"execution\_count": 4,

"metadata": {},

"outputs": [],

"source": [

"#First remove any movies that had a $0 domestic gross.\n",

"imdb\_budgets\_df = imdb\_budgets\_df[imdb\_budgets\_df['Domestic Gross'] !=0]"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Previewing the head of each dataframe so we know what data we are working with."

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Movie</th>\n",

" <th>Year</th>\n",

" <th>IMDb</th>\n",

" <th>Rating</th>\n",

" <th>Runtime</th>\n",

" <th>Genre</th>\n",

" <th>Release Date</th>\n",

" <th>Production Budget</th>\n",

" <th>Domestic Gross</th>\n",

" <th>Worldwide Gross</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>Avengers: Endgame</td>\n",

" <td>2019</td>\n",

" <td>8.4</td>\n",

" <td>PG-13</td>\n",

" <td>181</td>\n",

" <td>Action, Adventure, Drama</td>\n",

" <td>Apr 23, 2019</td>\n",

" <td>400000000</td>\n",

" <td>858373000</td>\n",

" <td>2797800564</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>Avatar</td>\n",

" <td>2009</td>\n",

" <td>7.8</td>\n",

" <td>PG-13</td>\n",

" <td>162</td>\n",

" <td>Action, Adventure, Fantasy</td>\n",

" <td>Dec 17, 2009</td>\n",

" <td>237000000</td>\n",

" <td>760507625</td>\n",

" <td>2788701337</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>Black Panther</td>\n",

" <td>2018</td>\n",

" <td>7.3</td>\n",

" <td>PG-13</td>\n",

" <td>134</td>\n",

" <td>Action, Adventure, Sci-Fi</td>\n",

" <td>Feb 13, 2018</td>\n",

" <td>200000000</td>\n",

" <td>700059566</td>\n",

" <td>1346103376</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>Avengers: Infinity War</td>\n",

" <td>2018</td>\n",

" <td>8.4</td>\n",

" <td>PG-13</td>\n",

" <td>149</td>\n",

" <td>Action, Adventure, Sci-Fi</td>\n",

" <td>Apr 25, 2018</td>\n",

" <td>300000000</td>\n",

" <td>678815482</td>\n",

" <td>2048359754</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>Titanic</td>\n",

" <td>1997</td>\n",

" <td>7.8</td>\n",

" <td>PG-13</td>\n",

" <td>194</td>\n",

" <td>Drama, Romance</td>\n",

" <td>Dec 18, 1997</td>\n",

" <td>200000000</td>\n",

" <td>659363944</td>\n",

" <td>2208208395</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Movie Year IMDb Rating Runtime \\\n",

"0 Avengers: Endgame 2019 8.4 PG-13 181 \n",

"1 Avatar 2009 7.8 PG-13 162 \n",

"2 Black Panther 2018 7.3 PG-13 134 \n",

"3 Avengers: Infinity War 2018 8.4 PG-13 149 \n",

"4 Titanic 1997 7.8 PG-13 194 \n",

"\n",

" Genre Release Date Production Budget \\\n",

"0 Action, Adventure, Drama Apr 23, 2019 400000000 \n",

"1 Action, Adventure, Fantasy Dec 17, 2009 237000000 \n",

"2 Action, Adventure, Sci-Fi Feb 13, 2018 200000000 \n",

"3 Action, Adventure, Sci-Fi Apr 25, 2018 300000000 \n",

"4 Drama, Romance Dec 18, 1997 200000000 \n",

"\n",

" Domestic Gross Worldwide Gross \n",

"0 858373000 2797800564 \n",

"1 760507625 2788701337 \n",

"2 700059566 1346103376 \n",

"3 678815482 2048359754 \n",

"4 659363944 2208208395 "

]

},

"execution\_count": 5,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"imdb\_budgets\_df.head()"

]

},

{

"cell\_type": "code",

"execution\_count": 6,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>movie</th>\n",

" <th>release\_date</th>\n",

" <th>release\_month</th>\n",

" <th>release\_day</th>\n",

" <th>release\_year</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>Metropolis</td>\n",

" <td>1927-03-06</td>\n",

" <td>March</td>\n",

" <td>Sunday</td>\n",

" <td>1927</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>Dr. Mabuse, the Gambler</td>\n",

" <td>1927-08-08</td>\n",

" <td>August</td>\n",

" <td>Monday</td>\n",

" <td>1927</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>The Unknown</td>\n",

" <td>1927-06-03</td>\n",

" <td>June</td>\n",

" <td>Friday</td>\n",

" <td>1927</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>The Jazz Singer</td>\n",

" <td>1927-10-06</td>\n",

" <td>October</td>\n",

" <td>Thursday</td>\n",

" <td>1927</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>Chicago</td>\n",

" <td>1927-12-23</td>\n",

" <td>December</td>\n",

" <td>Friday</td>\n",

" <td>1927</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" movie release\_date release\_month release\_day \\\n",

"0 Metropolis 1927-03-06 March Sunday \n",

"1 Dr. Mabuse, the Gambler 1927-08-08 August Monday \n",

"2 The Unknown 1927-06-03 June Friday \n",

"3 The Jazz Singer 1927-10-06 October Thursday \n",

"4 Chicago 1927-12-23 December Friday \n",

"\n",

" release\_year \n",

"0 1927 \n",

"1 1927 \n",

"2 1927 \n",

"3 1927 \n",

"4 1927 "

]

},

"execution\_count": 6,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"movie\_dates\_df.head()"

]

},

{

"cell\_type": "code",

"execution\_count": 7,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>title</th>\n",

" <th>max\_theaters</th>\n",

" <th>year</th>\n",

" <th>total\_dom\_gross($)</th>\n",

" <th>studio</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>The Lion King</td>\n",

" <td>4802</td>\n",

" <td>2019</td>\n",

" <td>543638043</td>\n",

" <td>Disney</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>Avengers: Endgame</td>\n",

" <td>4662</td>\n",

" <td>2019</td>\n",

" <td>858373000</td>\n",

" <td>Disney</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>Spider-Man: Far from Home</td>\n",

" <td>4634</td>\n",

" <td>2019</td>\n",

" <td>390532085</td>\n",

" <td>Sony</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>Toy Story 4</td>\n",

" <td>4575</td>\n",

" <td>2019</td>\n",

" <td>434038008</td>\n",

" <td>Disney</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>It Chapter Two</td>\n",

" <td>4570</td>\n",

" <td>2019</td>\n",

" <td>211593228</td>\n",

" <td>Warner Bros.</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" title max\_theaters year total\_dom\_gross($) \\\n",

"0 The Lion King 4802 2019 543638043 \n",

"1 Avengers: Endgame 4662 2019 858373000 \n",

"2 Spider-Man: Far from Home 4634 2019 390532085 \n",

"3 Toy Story 4 4575 2019 434038008 \n",

"4 It Chapter Two 4570 2019 211593228 \n",

"\n",

" studio \n",

"0 Disney \n",

"1 Disney \n",

"2 Sony \n",

"3 Disney \n",

"4 Warner Bros. "

]

},

"execution\_count": 7,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"theaters\_df.head()"

]

},

{

"cell\_type": "code",

"execution\_count": 8,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Movie</th>\n",

" <th>Year</th>\n",

" <th>value</th>\n",

" <th>Release Date</th>\n",

" <th>Production Budget</th>\n",

" <th>Domestic Gross</th>\n",

" <th>Worldwide Gross</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>Avengers: Endgame</td>\n",

" <td>2019</td>\n",

" <td>Robert Downey Jr.</td>\n",

" <td>Apr 23, 2019</td>\n",

" <td>400000000</td>\n",

" <td>858373000</td>\n",

" <td>2797800564</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>Avengers: Endgame</td>\n",

" <td>2019</td>\n",

" <td>Chris Evans</td>\n",

" <td>Apr 23, 2019</td>\n",

" <td>400000000</td>\n",

" <td>858373000</td>\n",

" <td>2797800564</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>Avengers: Endgame</td>\n",

" <td>2019</td>\n",

" <td>Mark Ruffalo</td>\n",

" <td>Apr 23, 2019</td>\n",

" <td>400000000</td>\n",

" <td>858373000</td>\n",

" <td>2797800564</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>Avengers: Endgame</td>\n",

" <td>2019</td>\n",

" <td>Chris Hemsworth</td>\n",

" <td>Apr 23, 2019</td>\n",

" <td>400000000</td>\n",

" <td>858373000</td>\n",

" <td>2797800564</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>Avatar</td>\n",

" <td>2009</td>\n",

" <td>Sam Worthington</td>\n",

" <td>Dec 17, 2009</td>\n",

" <td>237000000</td>\n",

" <td>760507625</td>\n",

" <td>2788701337</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Movie Year value Release Date \\\n",

"0 Avengers: Endgame 2019 Robert Downey Jr. Apr 23, 2019 \n",

"1 Avengers: Endgame 2019 Chris Evans Apr 23, 2019 \n",

"2 Avengers: Endgame 2019 Mark Ruffalo Apr 23, 2019 \n",

"3 Avengers: Endgame 2019 Chris Hemsworth Apr 23, 2019 \n",

"4 Avatar 2009 Sam Worthington Dec 17, 2009 \n",

"\n",

" Production Budget Domestic Gross Worldwide Gross \n",

"0 400000000 858373000 2797800564 \n",

"1 400000000 858373000 2797800564 \n",

"2 400000000 858373000 2797800564 \n",

"3 400000000 858373000 2797800564 \n",

"4 237000000 760507625 2788701337 "

]

},

"execution\_count": 8,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"actors\_df.head()"

]

},

{

"cell\_type": "code",

"execution\_count": 9,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Movie</th>\n",

" <th>Year</th>\n",

" <th>value</th>\n",

" <th>Release Date</th>\n",

" <th>Production Budget</th>\n",

" <th>Domestic Gross</th>\n",

" <th>Worldwide Gross</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>Avengers: Endgame</td>\n",

" <td>2019</td>\n",

" <td>Joe Russo</td>\n",

" <td>Apr 23, 2019</td>\n",

" <td>400000000</td>\n",

" <td>858373000</td>\n",

" <td>2797800564</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>Avengers: Endgame</td>\n",

" <td>2019</td>\n",

" <td>Anthony Russo</td>\n",

" <td>Apr 23, 2019</td>\n",

" <td>400000000</td>\n",

" <td>858373000</td>\n",

" <td>2797800564</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>Avatar</td>\n",

" <td>2009</td>\n",

" <td>James Cameron</td>\n",

" <td>Dec 17, 2009</td>\n",

" <td>237000000</td>\n",

" <td>760507625</td>\n",

" <td>2788701337</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>Black Panther</td>\n",

" <td>2018</td>\n",

" <td>Ryan Coogler</td>\n",

" <td>Feb 13, 2018</td>\n",

" <td>200000000</td>\n",

" <td>700059566</td>\n",

" <td>1346103376</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>Avengers: Infinity War</td>\n",

" <td>2018</td>\n",

" <td>Joe Russo</td>\n",

" <td>Apr 25, 2018</td>\n",

" <td>300000000</td>\n",

" <td>678815482</td>\n",

" <td>2048359754</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Movie Year value Release Date \\\n",

"0 Avengers: Endgame 2019 Joe Russo Apr 23, 2019 \n",

"1 Avengers: Endgame 2019 Anthony Russo Apr 23, 2019 \n",

"2 Avatar 2009 James Cameron Dec 17, 2009 \n",

"3 Black Panther 2018 Ryan Coogler Feb 13, 2018 \n",

"4 Avengers: Infinity War 2018 Joe Russo Apr 25, 2018 \n",

"\n",

" Production Budget Domestic Gross Worldwide Gross \n",

"0 400000000 858373000 2797800564 \n",

"1 400000000 858373000 2797800564 \n",

"2 237000000 760507625 2788701337 \n",

"3 200000000 700059566 1346103376 \n",

"4 300000000 678815482 2048359754 "

]

},

"execution\_count": 9,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"directors\_df.head()"

]

},

{

"cell\_type": "code",

"execution\_count": 10,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Movie</th>\n",

" <th>Year</th>\n",

" <th>IMDb</th>\n",

" <th>Rating</th>\n",

" <th>Runtime</th>\n",

" <th>Genre</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>Star Wars: Episode VII - The Force Awakens</td>\n",

" <td>2015</td>\n",

" <td>7.9</td>\n",

" <td>PG-13</td>\n",

" <td>138</td>\n",

" <td>Action, Adventure, Sci-Fi</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>Avengers: Endgame</td>\n",

" <td>2019</td>\n",

" <td>8.4</td>\n",

" <td>PG-13</td>\n",

" <td>181</td>\n",

" <td>Action, Adventure, Drama</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>Avatar</td>\n",

" <td>2009</td>\n",

" <td>7.8</td>\n",

" <td>PG-13</td>\n",

" <td>162</td>\n",

" <td>Action, Adventure, Fantasy</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>Black Panther</td>\n",

" <td>2018</td>\n",

" <td>7.3</td>\n",

" <td>PG-13</td>\n",

" <td>134</td>\n",

" <td>Action, Adventure, Sci-Fi</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>Avengers: Infinity War</td>\n",

" <td>2018</td>\n",

" <td>8.4</td>\n",

" <td>PG-13</td>\n",

" <td>149</td>\n",

" <td>Action, Adventure, Sci-Fi</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Movie Year IMDb Rating Runtime \\\n",

"0 Star Wars: Episode VII - The Force Awakens 2015 7.9 PG-13 138 \n",

"1 Avengers: Endgame 2019 8.4 PG-13 181 \n",

"2 Avatar 2009 7.8 PG-13 162 \n",

"3 Black Panther 2018 7.3 PG-13 134 \n",

"4 Avengers: Infinity War 2018 8.4 PG-13 149 \n",

"\n",

" Genre \n",

"0 Action, Adventure, Sci-Fi \n",

"1 Action, Adventure, Drama \n",

"2 Action, Adventure, Fantasy \n",

"3 Action, Adventure, Sci-Fi \n",

"4 Action, Adventure, Sci-Fi "

]

},

"execution\_count": 10,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"imdb\_base\_df.head()"

]

},

{

"cell\_type": "code",

"execution\_count": 11,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>title</th>\n",

" <th>studio</th>\n",

" <th>domestic\_gross</th>\n",

" <th>foreign\_gross</th>\n",

" <th>year</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>Toy Story 3</td>\n",

" <td>Buena Vista</td>\n",

" <td>415000000.0</td>\n",

" <td>652000000</td>\n",

" <td>2010</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>Alice in Wonderland (2010)</td>\n",

" <td>Buena Vista</td>\n",

" <td>334200000.0</td>\n",

" <td>691300000</td>\n",

" <td>2010</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>Harry Potter and the Deathly Hallows Part 1</td>\n",

" <td>WB</td>\n",

" <td>296000000.0</td>\n",

" <td>664300000</td>\n",

" <td>2010</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>Inception</td>\n",

" <td>WB</td>\n",

" <td>292600000.0</td>\n",

" <td>535700000</td>\n",

" <td>2010</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>Shrek Forever After</td>\n",

" <td>Pixar/Dreamworks</td>\n",

" <td>238700000.0</td>\n",

" <td>513900000</td>\n",

" <td>2010</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" title studio \\\n",

"0 Toy Story 3 Buena Vista \n",

"1 Alice in Wonderland (2010) Buena Vista \n",

"2 Harry Potter and the Deathly Hallows Part 1 WB \n",

"3 Inception WB \n",

"4 Shrek Forever After Pixar/Dreamworks \n",

"\n",

" domestic\_gross foreign\_gross year \n",

"0 415000000.0 652000000 2010 \n",

"1 334200000.0 691300000 2010 \n",

"2 296000000.0 664300000 2010 \n",

"3 292600000.0 535700000 2010 \n",

"4 238700000.0 513900000 2010 "

]

},

"execution\_count": 11,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"studio\_df.head()"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"# Question 1: What are the most profitable movies and how much should you spend?\n",

"\n",

"Let's calculate profit and profit margin for each of the movies in `imdb\_budgets\_df` dataframe and add those as new columns. \n",

"\n",

"Here, we'll define profit as `Worldwide Gross`-`Production Budget`.\n",

"\n",

"It will also be beneficial in our analysis to have uniformity when discussing movie budgets and profits so we will also create an adjusted budget and adjusted profit column to account for inflation.\n",

"\n",

"We will use an average inflation rate of 3.22%."

]

},

{

"cell\_type": "code",

"execution\_count": 12,

"metadata": {},

"outputs": [],

"source": [

"imdb\_budgets\_df['Profit'] = imdb\_budgets\_df['Worldwide Gross'] - imdb\_budgets\_df['Production Budget']\n",

"\n",

"imdb\_budgets\_df['Profit\_Margin'] = (imdb\_budgets\_df['Worldwide Gross'] - \n",

" imdb\_budgets\_df['Production Budget'])/imdb\_budgets\_df['Worldwide Gross']"

]

},

{

"cell\_type": "code",

"execution\_count": 13,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Movie</th>\n",

" <th>Year</th>\n",

" <th>IMDb</th>\n",

" <th>Rating</th>\n",

" <th>Runtime</th>\n",

" <th>Genre</th>\n",

" <th>Release Date</th>\n",

" <th>Production Budget</th>\n",

" <th>Domestic Gross</th>\n",

" <th>Worldwide Gross</th>\n",

" <th>Profit</th>\n",

" <th>Profit\_Margin</th>\n",

" <th>Adjusted\_Budget</th>\n",

" <th>Adjusted\_Profit</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>Avengers: Endgame</td>\n",

" <td>2019</td>\n",

" <td>8.40</td>\n",

" <td>PG-13</td>\n",

" <td>181</td>\n",

" <td>Action, Adventure, Drama</td>\n",

" <td>Apr 23, 2019</td>\n",

" <td>400000000</td>\n",

" <td>858373000</td>\n",

" <td>2797800564</td>\n",

" <td>2397800564</td>\n",

" <td>0.86</td>\n",

" <td>412880000.00</td>\n",

" <td>2475009742.16</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>Avatar</td>\n",

" <td>2009</td>\n",

" <td>7.80</td>\n",

" <td>PG-13</td>\n",

" <td>162</td>\n",

" <td>Action, Adventure, Fantasy</td>\n",

" <td>Dec 17, 2009</td>\n",

" <td>237000000</td>\n",

" <td>760507625</td>\n",

" <td>2788701337</td>\n",

" <td>2551701337</td>\n",

" <td>0.92</td>\n",

" <td>320945400.00</td>\n",

" <td>3455513950.57</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>Black Panther</td>\n",

" <td>2018</td>\n",

" <td>7.30</td>\n",

" <td>PG-13</td>\n",

" <td>134</td>\n",

" <td>Action, Adventure, Sci-Fi</td>\n",

" <td>Feb 13, 2018</td>\n",

" <td>200000000</td>\n",

" <td>700059566</td>\n",

" <td>1346103376</td>\n",

" <td>1146103376</td>\n",

" <td>0.85</td>\n",

" <td>212880000.00</td>\n",

" <td>1219912433.41</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>Avengers: Infinity War</td>\n",

" <td>2018</td>\n",

" <td>8.40</td>\n",

" <td>PG-13</td>\n",

" <td>149</td>\n",

" <td>Action, Adventure, Sci-Fi</td>\n",

" <td>Apr 25, 2018</td>\n",

" <td>300000000</td>\n",

" <td>678815482</td>\n",

" <td>2048359754</td>\n",

" <td>1748359754</td>\n",

" <td>0.85</td>\n",

" <td>319320000.00</td>\n",

" <td>1860954122.16</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>Titanic</td>\n",

" <td>1997</td>\n",

" <td>7.80</td>\n",

" <td>PG-13</td>\n",

" <td>194</td>\n",

" <td>Drama, Romance</td>\n",

" <td>Dec 18, 1997</td>\n",

" <td>200000000</td>\n",

" <td>659363944</td>\n",

" <td>2208208395</td>\n",

" <td>2008208395</td>\n",

" <td>0.91</td>\n",

" <td>348120000.00</td>\n",

" <td>3495487532.34</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Movie Year IMDb Rating Runtime \\\n",

"0 Avengers: Endgame 2019 8.40 PG-13 181 \n",

"1 Avatar 2009 7.80 PG-13 162 \n",

"2 Black Panther 2018 7.30 PG-13 134 \n",

"3 Avengers: Infinity War 2018 8.40 PG-13 149 \n",

"4 Titanic 1997 7.80 PG-13 194 \n",

"\n",

" Genre Release Date Production Budget \\\n",

"0 Action, Adventure, Drama Apr 23, 2019 400000000 \n",

"1 Action, Adventure, Fantasy Dec 17, 2009 237000000 \n",

"2 Action, Adventure, Sci-Fi Feb 13, 2018 200000000 \n",

"3 Action, Adventure, Sci-Fi Apr 25, 2018 300000000 \n",

"4 Drama, Romance Dec 18, 1997 200000000 \n",

"\n",

" Domestic Gross Worldwide Gross Profit Profit\_Margin \\\n",

"0 858373000 2797800564 2397800564 0.86 \n",

"1 760507625 2788701337 2551701337 0.92 \n",

"2 700059566 1346103376 1146103376 0.85 \n",

"3 678815482 2048359754 1748359754 0.85 \n",

"4 659363944 2208208395 2008208395 0.91 \n",

"\n",

" Adjusted\_Budget Adjusted\_Profit \n",

"0 412880000.00 2475009742.16 \n",

"1 320945400.00 3455513950.57 \n",

"2 212880000.00 1219912433.41 \n",

"3 319320000.00 1860954122.16 \n",

"4 348120000.00 3495487532.34 "

]

},

"execution\_count": 13,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"imdb\_budgets\_df['Adjusted\_Budget'] = ((((2020-imdb\_budgets\_df['Year'])\*.0322)+1)\*\n",

" imdb\_budgets\_df['Production Budget'])\n",

"\n",

"#Suppressing Scienific Notation\n",

"pd.options.display.float\_format = '{:.2f}'.format\n",

"\n",

"imdb\_budgets\_df['Adjusted\_Profit'] = (((2020-imdb\_budgets\_df['Year'])\*.0322)+1)\*imdb\_budgets\_df['Profit']\n",

"imdb\_budgets\_df.head()"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"For this question we are specifically looking at profitable movies. We'll create a separate dataframe called `profitable\_movies\_df` where the `Profit` column is greater than 0. We will then sort by `Adjusted\_Profit` to rank movies in terms of profitability."

]

},

{

"cell\_type": "code",

"execution\_count": 14,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>index</th>\n",

" <th>Movie</th>\n",

" <th>Year</th>\n",

" <th>IMDb</th>\n",

" <th>Rating</th>\n",

" <th>Runtime</th>\n",

" <th>Genre</th>\n",

" <th>Release Date</th>\n",

" <th>Production Budget</th>\n",

" <th>Domestic Gross</th>\n",

" <th>Worldwide Gross</th>\n",

" <th>Profit</th>\n",

" <th>Profit\_Margin</th>\n",

" <th>Adjusted\_Budget</th>\n",

" <th>Adjusted\_Profit</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>4</td>\n",

" <td>Titanic</td>\n",

" <td>1997</td>\n",

" <td>7.80</td>\n",

" <td>PG-13</td>\n",

" <td>194</td>\n",

" <td>Drama, Romance</td>\n",

" <td>Dec 18, 1997</td>\n",

" <td>200000000</td>\n",

" <td>659363944</td>\n",

" <td>2208208395</td>\n",

" <td>2008208395</td>\n",

" <td>0.91</td>\n",

" <td>348120000.00</td>\n",

" <td>3495487532.34</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>1</td>\n",

" <td>Avatar</td>\n",

" <td>2009</td>\n",

" <td>7.80</td>\n",

" <td>PG-13</td>\n",

" <td>162</td>\n",

" <td>Action, Adventure, Fantasy</td>\n",

" <td>Dec 17, 2009</td>\n",

" <td>237000000</td>\n",

" <td>760507625</td>\n",

" <td>2788701337</td>\n",

" <td>2551701337</td>\n",

" <td>0.92</td>\n",

" <td>320945400.00</td>\n",

" <td>3455513950.57</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>0</td>\n",

" <td>Avengers: Endgame</td>\n",

" <td>2019</td>\n",

" <td>8.40</td>\n",

" <td>PG-13</td>\n",

" <td>181</td>\n",

" <td>Action, Adventure, Drama</td>\n",

" <td>Apr 23, 2019</td>\n",

" <td>400000000</td>\n",

" <td>858373000</td>\n",

" <td>2797800564</td>\n",

" <td>2397800564</td>\n",

" <td>0.86</td>\n",

" <td>412880000.00</td>\n",

" <td>2475009742.16</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>3</td>\n",

" <td>Avengers: Infinity War</td>\n",

" <td>2018</td>\n",

" <td>8.40</td>\n",

" <td>PG-13</td>\n",

" <td>149</td>\n",

" <td>Action, Adventure, Sci-Fi</td>\n",

" <td>Apr 25, 2018</td>\n",

" <td>300000000</td>\n",

" <td>678815482</td>\n",

" <td>2048359754</td>\n",

" <td>1748359754</td>\n",

" <td>0.85</td>\n",

" <td>319320000.00</td>\n",

" <td>1860954122.16</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>28</td>\n",

" <td>Jurassic Park</td>\n",

" <td>1993</td>\n",

" <td>8.10</td>\n",

" <td>PG-13</td>\n",

" <td>127</td>\n",

" <td>Action, Adventure, Sci-Fi</td>\n",

" <td>Jun 11, 1993</td>\n",

" <td>63000000</td>\n",

" <td>402523348</td>\n",

" <td>1045627627</td>\n",

" <td>982627627</td>\n",

" <td>0.94</td>\n",

" <td>117772200.00</td>\n",

" <td>1836924085.91</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" index Movie Year IMDb Rating Runtime \\\n",

"0 4 Titanic 1997 7.80 PG-13 194 \n",

"1 1 Avatar 2009 7.80 PG-13 162 \n",

"2 0 Avengers: Endgame 2019 8.40 PG-13 181 \n",

"3 3 Avengers: Infinity War 2018 8.40 PG-13 149 \n",

"4 28 Jurassic Park 1993 8.10 PG-13 127 \n",

"\n",

" Genre Release Date Production Budget \\\n",

"0 Drama, Romance Dec 18, 1997 200000000 \n",

"1 Action, Adventure, Fantasy Dec 17, 2009 237000000 \n",

"2 Action, Adventure, Drama Apr 23, 2019 400000000 \n",

"3 Action, Adventure, Sci-Fi Apr 25, 2018 300000000 \n",

"4 Action, Adventure, Sci-Fi Jun 11, 1993 63000000 \n",

"\n",

" Domestic Gross Worldwide Gross Profit Profit\_Margin \\\n",

"0 659363944 2208208395 2008208395 0.91 \n",

"1 760507625 2788701337 2551701337 0.92 \n",

"2 858373000 2797800564 2397800564 0.86 \n",

"3 678815482 2048359754 1748359754 0.85 \n",

"4 402523348 1045627627 982627627 0.94 \n",

"\n",

" Adjusted\_Budget Adjusted\_Profit \n",

"0 348120000.00 3495487532.34 \n",

"1 320945400.00 3455513950.57 \n",

"2 412880000.00 2475009742.16 \n",

"3 319320000.00 1860954122.16 \n",

"4 117772200.00 1836924085.91 "

]

},

"execution\_count": 14,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"profitable\_movies\_df = imdb\_budgets\_df.loc[imdb\_budgets\_df['Profit'] > 0]\n",

"profitable\_ranked\_df = profitable\_movies\_df.sort\_values(by=['Adjusted\_Profit'], ascending=False)\n",

"profitable\_ranked\_df.reset\_index(inplace=True) #Modify the DataFrame in place (do not create a new object).\n",

"profitable\_ranked\_df.head()"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Now that we've got our profitable movie data, let's take a look at adjusted profit versus adjusted budget for each of the movies in the dataframe."

]

},

{

"cell\_type": "code",

"execution\_count": 15,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1008x504 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"ax1 = sns.lmplot(x='Adjusted\_Budget', y='Adjusted\_Profit', data=profitable\_ranked\_df, height=7, aspect=2)\n",

"plt.xlabel('Budget Adjusted for Inflation (Millions of Dollars)', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"#setting x-axis label \n",

"plt.ticklabel\_format(axis='x', style='sci', scilimits=(6,6))\n",

"plt.ylabel('Profit Adjusted for Inflation (Billions of Dollars)',fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('Adjusted Budget vs Adjusted Profit for Profitable Movies', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.savefig('BudgetVProfit', dpi = 300);"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"This scatter plot is helpful in beginning to understand how much money should be budgeted for a movie. The positive trend line indicates that an increase in the budget will result in an increase in profit.\n",

"\n",

"Let's take a look at the most successful movies so that we can get a better idea of what the budget should be."

]

},

{

"cell\_type": "code",

"execution\_count": 16,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1080x864 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"plt.figure(figsize=(15,12))\n",

"sns.barplot(x=profitable\_ranked\_df.loc[0:25, 'Movie'],y=profitable\_ranked\_df.loc[0:25, 'Adjusted\_Profit'], \n",

" color='mediumspringgreen', label='Profit', ci=None)\n",

"sns.barplot(x=profitable\_ranked\_df.loc[0:25, 'Movie'],y=profitable\_ranked\_df.loc[0:25, 'Adjusted\_Budget'], \n",

" color='black', label='Budget', ci=None)\n",

"plt.xlabel('Movie', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title(\"Profits and Budgets for the 25 Most Profitable Movies\", fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.ylabel('Adjusted Profit (Billions of Dollars)', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.xticks(rotation=35, horizontalalignment='right', fontsize=15)\n",

"plt.legend(loc='best', fontsize=15)\n",

"plt.savefig('ProfitBudgetTop25', dpi=300);"

]

},

{

"cell\_type": "code",

"execution\_count": 17,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"count 2836.00\n",

"mean 60689139.20\n",

"std 63199464.86\n",

"min 10606.40\n",

"25% 16608850.00\n",

"50% 38684100.00\n",

"75% 82247150.00\n",

"max 488834200.00\n",

"Name: Adjusted\_Budget, dtype: float64"

]

},

"execution\_count": 17,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"profitable\_movies\_df['Adjusted\_Budget'].describe()"

]

},

{

"cell\_type": "code",

"execution\_count": 18,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"count 25.00\n",

"mean 242777774.40\n",

"std 80698866.89\n",

"min 106064000.00\n",

"25% 180635000.00\n",

"50% 225760000.00\n",

"75% 282960000.00\n",

"max 423765000.00\n",

"Name: Adjusted\_Budget, dtype: float64"

]

},

"execution\_count": 18,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"profitable\_movies\_df.loc[0:24, 'Adjusted\_Budget'].describe()"

]

},

{

"cell\_type": "code",

"execution\_count": 19,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"count 2836.00\n",

"mean 0.62\n",

"std 0.24\n",

"min 0.00\n",

"25% 0.47\n",

"50% 0.67\n",

"75% 0.81\n",

"max 1.00\n",

"Name: Profit\_Margin, dtype: float64"

]

},

"execution\_count": 19,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"profitable\_movies\_df['Profit\_Margin'].describe()"

]

},

{

"cell\_type": "code",

"execution\_count": 20,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"count 25.00\n",

"mean 0.85\n",

"std 0.05\n",

"min 0.74\n",

"25% 0.81\n",

"50% 0.85\n",

"75% 0.87\n",

"max 0.93\n",

"Name: Profit\_Margin, dtype: float64"

]

},

"execution\_count": 20,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"profitable\_movies\_df.loc[0:24, 'Profit\_Margin'].describe()"

]

},

{

"cell\_type": "code",

"execution\_count": 21,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"2041"

]

},

"execution\_count": 21,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"len(profitable\_ranked\_df.loc[profitable\_ranked\_df['Profit\_Margin'] > 0.5])"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Clearly the most successful 25 movies have both incredible profits and profit margins. Titanic (1997), Avatar, and Avengers: Endgame are the most successful movies in terms of sheer profit. \n",

"\n",

"So how do we know what to spend? We need to think about what sort of profit margin we want to see. 2043 out of 2841 total profitable movies have a profit margin over 50%. That's good news as it indicates that we can be more aggressive in choosing a threshold for the profit margin. The top 25 movies have a median profit margin of 84.9% with a median budget of \\\\$225,760,000. When looking at all of our profitable movies, the profit margin drops significantly to 67.1\\% and the budget drops significantly to \\\\$38,676,000. We use the median to describe our data here as the mean will be skewed by outlier data.\n",

"\n",

"Let's filter the data with a profit margin of 75% or greater and a budget greater than $38,676,000."

]

},

{

"cell\_type": "code",

"execution\_count": 23,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"374"

]

},

"execution\_count": 23,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"filtered\_df = profitable\_ranked\_df.loc[(profitable\_ranked\_df['Profit\_Margin'] >= 0.75) & \n",

" (profitable\_ranked\_df['Adjusted\_Budget'] > 38676000)]\n",

"len(filtered\_df)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"After filtering we still have 374 movies left upon which to draw conclusions."

]

},

{

"cell\_type": "code",

"execution\_count": 32,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1008x504 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"ax2 = sns.lmplot(x='Adjusted\_Budget', y='Profit\_Margin', data=filtered\_df, height=7, aspect=2)\n",

"plt.xlabel('Adjusted Budget (Millions of Dollars)', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.ticklabel\_format(axis='x', style='sci', scilimits=(6,6))\n",

"plt.ylabel('Profit Margin', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('Adjusted Budget vs Profit Margin', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.savefig('BudgetVMargin', dpi=300);"

]

},

{

"cell\_type": "code",

"execution\_count": 25,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>index</th>\n",

" <th>Year</th>\n",

" <th>IMDb</th>\n",

" <th>Runtime</th>\n",

" <th>Production Budget</th>\n",

" <th>Domestic Gross</th>\n",

" <th>Worldwide Gross</th>\n",

" <th>Profit</th>\n",

" <th>Profit\_Margin</th>\n",

" <th>Adjusted\_Budget</th>\n",

" <th>Adjusted\_Profit</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>count</th>\n",

" <td>374.00</td>\n",

" <td>374.00</td>\n",

" <td>374.00</td>\n",

" <td>374.00</td>\n",

" <td>374.00</td>\n",

" <td>374.00</td>\n",

" <td>374.00</td>\n",

" <td>374.00</td>\n",

" <td>374.00</td>\n",

" <td>374.00</td>\n",

" <td>374.00</td>\n",

" </tr>\n",

" <tr>\n",

" <th>mean</th>\n",

" <td>391.53</td>\n",

" <td>2004.97</td>\n",

" <td>7.01</td>\n",

" <td>118.60</td>\n",

" <td>77814178.13</td>\n",

" <td>193378841.67</td>\n",

" <td>484994903.63</td>\n",

" <td>407180725.50</td>\n",

" <td>0.83</td>\n",

" <td>105858522.51</td>\n",

" <td>562879114.94</td>\n",

" </tr>\n",

" <tr>\n",

" <th>std</th>\n",

" <td>378.20</td>\n",

" <td>10.81</td>\n",

" <td>0.90</td>\n",

" <td>24.02</td>\n",

" <td>57570152.51</td>\n",

" <td>127088965.57</td>\n",

" <td>377690264.14</td>\n",

" <td>329994078.69</td>\n",

" <td>0.05</td>\n",

" <td>66272237.80</td>\n",

" <td>413114307.71</td>\n",

" </tr>\n",

" <tr>\n",

" <th>min</th>\n",

" <td>0.00</td>\n",

" <td>1956.00</td>\n",

" <td>3.30</td>\n",

" <td>79.00</td>\n",

" <td>13500000.00</td>\n",

" <td>19019882.00</td>\n",

" <td>69995385.00</td>\n",

" <td>54995385.00</td>\n",

" <td>0.75</td>\n",

" <td>38685000.00</td>\n",

" <td>123209844.42</td>\n",

" </tr>\n",

" <tr>\n",

" <th>25%</th>\n",

" <td>111.25</td>\n",

" <td>1998.00</td>\n",

" <td>6.40</td>\n",

" <td>100.00</td>\n",

" <td>35000000.00</td>\n",

" <td>106948347.75</td>\n",

" <td>217288435.75</td>\n",

" <td>176354400.25</td>\n",

" <td>0.78</td>\n",

" <td>53471100.00</td>\n",

" <td>274861614.08</td>\n",

" </tr>\n",

" <tr>\n",

" <th>50%</th>\n",

" <td>279.50</td>\n",

" <td>2007.00</td>\n",

" <td>7.00</td>\n",

" <td>116.00</td>\n",

" <td>55000000.00</td>\n",

" <td>162801999.50</td>\n",

" <td>350937609.00</td>\n",

" <td>299062980.00</td>\n",

" <td>0.82</td>\n",

" <td>82249300.00</td>\n",

" <td>449229900.01</td>\n",

" </tr>\n",

" <tr>\n",

" <th>75%</th>\n",

" <td>550.50</td>\n",

" <td>2014.00</td>\n",

" <td>7.70</td>\n",

" <td>131.75</td>\n",

" <td>100000000.00</td>\n",

" <td>242081446.50</td>\n",

" <td>636084264.50</td>\n",

" <td>513979301.75</td>\n",

" <td>0.87</td>\n",

" <td>139654600.00</td>\n",

" <td>719591073.46</td>\n",

" </tr>\n",

" <tr>\n",

" <th>max</th>\n",

" <td>2424.00</td>\n",

" <td>2020.00</td>\n",

" <td>9.00</td>\n",

" <td>228.00</td>\n",

" <td>400000000.00</td>\n",

" <td>858373000.00</td>\n",

" <td>2797800564.00</td>\n",

" <td>2551701337.00</td>\n",

" <td>0.96</td>\n",

" <td>412880000.00</td>\n",

" <td>3495487532.34</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" index Year IMDb Runtime Production Budget Domestic Gross \\\n",

"count 374.00 374.00 374.00 374.00 374.00 374.00 \n",

"mean 391.53 2004.97 7.01 118.60 77814178.13 193378841.67 \n",

"std 378.20 10.81 0.90 24.02 57570152.51 127088965.57 \n",

"min 0.00 1956.00 3.30 79.00 13500000.00 19019882.00 \n",

"25% 111.25 1998.00 6.40 100.00 35000000.00 106948347.75 \n",

"50% 279.50 2007.00 7.00 116.00 55000000.00 162801999.50 \n",

"75% 550.50 2014.00 7.70 131.75 100000000.00 242081446.50 \n",

"max 2424.00 2020.00 9.00 228.00 400000000.00 858373000.00 \n",

"\n",

" Worldwide Gross Profit Profit\_Margin Adjusted\_Budget \\\n",

"count 374.00 374.00 374.00 374.00 \n",

"mean 484994903.63 407180725.50 0.83 105858522.51 \n",

"std 377690264.14 329994078.69 0.05 66272237.80 \n",

"min 69995385.00 54995385.00 0.75 38685000.00 \n",

"25% 217288435.75 176354400.25 0.78 53471100.00 \n",

"50% 350937609.00 299062980.00 0.82 82249300.00 \n",

"75% 636084264.50 513979301.75 0.87 139654600.00 \n",

"max 2797800564.00 2551701337.00 0.96 412880000.00 \n",

"\n",

" Adjusted\_Profit \n",

"count 374.00 \n",

"mean 562879114.94 \n",

"std 413114307.71 \n",

"min 123209844.42 \n",

"25% 274861614.08 \n",

"50% 449229900.01 \n",

"75% 719591073.46 \n",

"max 3495487532.34 "

]

},

"execution\_count": 25,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"filtered\_df.describe()"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"We examine the data in a scatter plot again to see if we can determine trends. Our data is much more spread out when comparing profit margin and budget. The trend line in this plot is negative which cautions against spending too much money as we may potentially hurt our profit margin. Looking at the filtered data, we have a median budget of $82,249,300 and a median profit margin of 81.9%. \n",

"\n",

"\*\*Question 1 Conclusion\*\*: We recommend that our Company should budget approximately $82,250,000 to make a movie. This should correlate with a profit margin above 80%. "

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"# Question 2: Which movie genres are most commonly produced and does quantity equate to higher net profits?"

]

},

{

"cell\_type": "code",

"execution\_count": 27,

"metadata": {},

"outputs": [],

"source": [

"#Create a genre table that separates each value in the genre column in their own rows.\n",

"imdb\_budgets\_df['Genre'] = imdb\_budgets\_df['Genre'].str.split(', ')\n",

"imdb\_budgets\_df1 = imdb\_budgets\_df['Genre'].apply(pd.Series)\n",

"\n",

"imdb\_budgets\_df2 = pd.merge(imdb\_budgets\_df, imdb\_budgets\_df1, right\_index = True, left\_index = True)\n",

"\n",

"imdb\_budgets\_df3 = imdb\_budgets\_df2.drop(['Genre'], axis = 1)\n",

"\n",

"genre\_budgets\_df = imdb\_budgets\_df3.melt(id\_vars=['Movie', 'Year'], value\_vars=[0, 1, 2] ,var\_name = ['X'])\n",

"genre\_budgets\_df = pd.merge(genre\_budgets\_df, imdb\_budgets\_df)\n",

"genre\_budgets\_df = genre\_budgets\_df.drop(['Genre', 'X'], axis=1)\n",

"genre\_budgets\_df = genre\_budgets\_df.drop\_duplicates()\n",

"genre\_budgets\_df = genre\_budgets\_df.rename(columns={'value': 'Genre'})\n",

"genre\_budgets\_df = genre\_budgets\_df.dropna()"

]

},

{

"cell\_type": "code",

"execution\_count": 28,

"metadata": {},

"outputs": [],

"source": [

"#Do a count of all movies grouped by genre.\n",

"m\_by\_genre = genre\_budgets\_df.groupby('Genre', as\_index=False)['Movie'].count().sort\_values(by='Movie', ascending=False)"

]

},

{

"cell\_type": "code",

"execution\_count": 29,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Genre</th>\n",

" <th>Movie</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>6</th>\n",

" <td>Drama</td>\n",

" <td>1876</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>Comedy</td>\n",

" <td>1444</td>\n",

" </tr>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>Action</td>\n",

" <td>1045</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>Adventure</td>\n",

" <td>834</td>\n",

" </tr>\n",

" <tr>\n",

" <th>5</th>\n",

" <td>Crime</td>\n",

" <td>689</td>\n",

" </tr>\n",

" <tr>\n",

" <th>15</th>\n",

" <td>Romance</td>\n",

" <td>622</td>\n",

" </tr>\n",

" <tr>\n",

" <th>18</th>\n",

" <td>Thriller</td>\n",

" <td>615</td>\n",

" </tr>\n",

" <tr>\n",

" <th>11</th>\n",

" <td>Horror</td>\n",

" <td>410</td>\n",

" </tr>\n",

" <tr>\n",

" <th>14</th>\n",

" <td>Mystery</td>\n",

" <td>356</td>\n",

" </tr>\n",

" <tr>\n",

" <th>16</th>\n",

" <td>Sci-Fi</td>\n",

" <td>330</td>\n",

" </tr>\n",

" <tr>\n",

" <th>8</th>\n",

" <td>Fantasy</td>\n",

" <td>324</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>Biography</td>\n",

" <td>294</td>\n",

" </tr>\n",

" <tr>\n",

" <th>7</th>\n",

" <td>Family</td>\n",

" <td>265</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>Animation</td>\n",

" <td>205</td>\n",

" </tr>\n",

" <tr>\n",

" <th>10</th>\n",

" <td>History</td>\n",

" <td>123</td>\n",

" </tr>\n",

" <tr>\n",

" <th>12</th>\n",

" <td>Music</td>\n",

" <td>109</td>\n",

" </tr>\n",

" <tr>\n",

" <th>17</th>\n",

" <td>Sport</td>\n",

" <td>100</td>\n",

" </tr>\n",

" <tr>\n",

" <th>19</th>\n",

" <td>War</td>\n",

" <td>68</td>\n",

" </tr>\n",

" <tr>\n",

" <th>13</th>\n",

" <td>Musical</td>\n",

" <td>39</td>\n",

" </tr>\n",

" <tr>\n",

" <th>20</th>\n",

" <td>Western</td>\n",

" <td>32</td>\n",

" </tr>\n",

" <tr>\n",

" <th>9</th>\n",

" <td>Film-Noir</td>\n",

" <td>6</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Genre Movie\n",

"6 Drama 1876\n",

"4 Comedy 1444\n",

"0 Action 1045\n",

"1 Adventure 834\n",

"5 Crime 689\n",

"15 Romance 622\n",

"18 Thriller 615\n",

"11 Horror 410\n",

"14 Mystery 356\n",

"16 Sci-Fi 330\n",

"8 Fantasy 324\n",

"3 Biography 294\n",

"7 Family 265\n",

"2 Animation 205\n",

"10 History 123\n",

"12 Music 109\n",

"17 Sport 100\n",

"19 War 68\n",

"13 Musical 39\n",

"20 Western 32\n",

"9 Film-Noir 6"

]

},

"execution\_count": 29,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"m\_by\_genre"

]

},

{

"cell\_type": "code",

"execution\_count": 31,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1008x504 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"#Plot the above findings.\n",

"plt.figure(figsize=(14,7))\n",

"ax3 = sns.barplot(x=m\_by\_genre['Movie'], y=m\_by\_genre['Genre'], palette='GnBu\_d')\n",

"plt.xlabel('Movie Count', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.ylabel('Genre', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('Movie Count By Genre', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.savefig('CountGenre', dpi=300);"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"We can see that drama, comedy, and action dominate the quantity of movie genres but does this necessarily mean these are the most profitable genres? In order to determine this we will once again group each genre but this time we are going to take a look at the average net profit for each."

]

},

{

"cell\_type": "code",

"execution\_count": 35,

"metadata": {},

"outputs": [],

"source": [

"#Once again group the movies by genre, showing the average net profit and profit margin for each.\n",

"p\_by\_genre = genre\_budgets\_df.groupby('Genre', as\_index=False)[['Adjusted\_Profit', 'Profit\_Margin']].median().sort\_values(by='Adjusted\_Profit', ascending=False)\n"

]

},

{

"cell\_type": "code",

"execution\_count": 36,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Genre</th>\n",

" <th>Adjusted\_Profit</th>\n",

" <th>Profit\_Margin</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>Animation</td>\n",

" <td>203606574.36</td>\n",

" <td>0.68</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>Adventure</td>\n",

" <td>123795016.96</td>\n",

" <td>0.61</td>\n",

" </tr>\n",

" <tr>\n",

" <th>16</th>\n",

" <td>Sci-Fi</td>\n",

" <td>76199115.79</td>\n",

" <td>0.60</td>\n",

" </tr>\n",

" <tr>\n",

" <th>7</th>\n",

" <td>Family</td>\n",

" <td>74621544.29</td>\n",

" <td>0.58</td>\n",

" </tr>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>Action</td>\n",

" <td>64332532.19</td>\n",

" <td>0.52</td>\n",

" </tr>\n",

" <tr>\n",

" <th>8</th>\n",

" <td>Fantasy</td>\n",

" <td>54057582.24</td>\n",

" <td>0.54</td>\n",

" </tr>\n",

" <tr>\n",

" <th>18</th>\n",

" <td>Thriller</td>\n",

" <td>47338952.53</td>\n",

" <td>0.60</td>\n",

" </tr>\n",

" <tr>\n",

" <th>13</th>\n",

" <td>Musical</td>\n",

" <td>46631897.60</td>\n",

" <td>0.65</td>\n",

" </tr>\n",

" <tr>\n",

" <th>14</th>\n",

" <td>Mystery</td>\n",

" <td>39634323.82</td>\n",

" <td>0.61</td>\n",

" </tr>\n",

" <tr>\n",

" <th>11</th>\n",

" <td>Horror</td>\n",

" <td>38963349.12</td>\n",

" <td>0.67</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>Comedy</td>\n",

" <td>33917454.39</td>\n",

" <td>0.55</td>\n",

" </tr>\n",

" <tr>\n",

" <th>15</th>\n",

" <td>Romance</td>\n",

" <td>26739545.09</td>\n",

" <td>0.57</td>\n",

" </tr>\n",

" <tr>\n",

" <th>10</th>\n",

" <td>History</td>\n",

" <td>23435554.73</td>\n",

" <td>0.40</td>\n",

" </tr>\n",

" <tr>\n",

" <th>6</th>\n",

" <td>Drama</td>\n",

" <td>23258412.08</td>\n",

" <td>0.50</td>\n",

" </tr>\n",

" <tr>\n",

" <th>12</th>\n",

" <td>Music</td>\n",

" <td>22774962.29</td>\n",

" <td>0.55</td>\n",

" </tr>\n",

" <tr>\n",

" <th>5</th>\n",

" <td>Crime</td>\n",

" <td>22752334.82</td>\n",

" <td>0.40</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>Biography</td>\n",

" <td>21750633.96</td>\n",

" <td>0.43</td>\n",

" </tr>\n",

" <tr>\n",

" <th>9</th>\n",

" <td>Film-Noir</td>\n",

" <td>18766783.04</td>\n",

" <td>0.81</td>\n",

" </tr>\n",

" <tr>\n",

" <th>19</th>\n",

" <td>War</td>\n",

" <td>18653512.63</td>\n",

" <td>0.37</td>\n",

" </tr>\n",

" <tr>\n",

" <th>17</th>\n",

" <td>Sport</td>\n",

" <td>17950554.99</td>\n",

" <td>0.35</td>\n",

" </tr>\n",

" <tr>\n",

" <th>20</th>\n",

" <td>Western</td>\n",

" <td>12037135.33</td>\n",

" <td>0.39</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Genre Adjusted\_Profit Profit\_Margin\n",

"2 Animation 203606574.36 0.68\n",

"1 Adventure 123795016.96 0.61\n",

"16 Sci-Fi 76199115.79 0.60\n",

"7 Family 74621544.29 0.58\n",

"0 Action 64332532.19 0.52\n",

"8 Fantasy 54057582.24 0.54\n",

"18 Thriller 47338952.53 0.60\n",

"13 Musical 46631897.60 0.65\n",

"14 Mystery 39634323.82 0.61\n",

"11 Horror 38963349.12 0.67\n",

"4 Comedy 33917454.39 0.55\n",

"15 Romance 26739545.09 0.57\n",

"10 History 23435554.73 0.40\n",

"6 Drama 23258412.08 0.50\n",

"12 Music 22774962.29 0.55\n",

"5 Crime 22752334.82 0.40\n",

"3 Biography 21750633.96 0.43\n",

"9 Film-Noir 18766783.04 0.81\n",

"19 War 18653512.63 0.37\n",

"17 Sport 17950554.99 0.35\n",

"20 Western 12037135.33 0.39"

]

},

"execution\_count": 36,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"p\_by\_genre"

]

},

{

"cell\_type": "code",

"execution\_count": 37,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1008x504 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"#Plot the above findings.\n",

"plt.figure(figsize=(14,7))\n",

"ax4 = sns.barplot(x=p\_by\_genre['Adjusted\_Profit'], y=p\_by\_genre['Genre'], palette='GnBu\_d')\n",

"plt.xlabel('Net Profit (Hundreds of Millions)', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.ylabel('Genre',fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('Net Profit By Genre', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.savefig('NetProfitGenre', dpi=300);"

]

},

{

"cell\_type": "code",

"execution\_count": 38,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1008x504 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"plt.figure(figsize=(14,7))\n",

"ax5 = sns.barplot(x=p\_by\_genre['Profit\_Margin'], y=p\_by\_genre['Genre'], palette='GnBu\_d')\n",

"plt.xlabel('Genre', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.ylabel('Profit Margin', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('Profit Margin By Genre', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.xlim(0.3, 0.85)\n",

"plt.savefig('ProfitMarginGenre', dpi=300);"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Interesting, although they are not the most commonly released genres; animation, adventure, and sci-fi typically have the most success in terms of median net profit. We can also see that Animation has a desirable profit margin along with horror and musicals. Note: although Film Noir leads with a .8+ profit margin this is based on 6 movies and has to be disregarded due to the small sample size. \n",

"\n",

"Lastly, of what percentage of the total net profit from all genres does each genre account?"

]

},

{

"cell\_type": "code",

"execution\_count": 39,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Genre</th>\n",

" <th>Adjusted\_Profit</th>\n",

" <th>Percent Total of Net Profit</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>Adventure</td>\n",

" <td>217335741708.40</td>\n",

" <td>18.07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>Action</td>\n",

" <td>178930045524.32</td>\n",

" <td>14.88</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>Comedy</td>\n",

" <td>151922895671.69</td>\n",

" <td>12.63</td>\n",

" </tr>\n",

" <tr>\n",

" <th>6</th>\n",

" <td>Drama</td>\n",

" <td>144990041873.71</td>\n",

" <td>12.05</td>\n",

" </tr>\n",

" <tr>\n",

" <th>16</th>\n",

" <td>Sci-Fi</td>\n",

" <td>70465612908.78</td>\n",

" <td>5.86</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>Animation</td>\n",

" <td>68720987812.40</td>\n",

" <td>5.71</td>\n",

" </tr>\n",

" <tr>\n",

" <th>18</th>\n",

" <td>Thriller</td>\n",

" <td>65442236225.98</td>\n",

" <td>5.44</td>\n",

" </tr>\n",

" <tr>\n",

" <th>8</th>\n",

" <td>Fantasy</td>\n",

" <td>54797139085.80</td>\n",

" <td>4.56</td>\n",

" </tr>\n",

" <tr>\n",

" <th>15</th>\n",

" <td>Romance</td>\n",

" <td>50510744180.92</td>\n",

" <td>4.20</td>\n",

" </tr>\n",

" <tr>\n",

" <th>7</th>\n",

" <td>Family</td>\n",

" <td>46040638020.14</td>\n",

" <td>3.83</td>\n",

" </tr>\n",

" <tr>\n",

" <th>5</th>\n",

" <td>Crime</td>\n",

" <td>45194406614.69</td>\n",

" <td>3.76</td>\n",

" </tr>\n",

" <tr>\n",

" <th>14</th>\n",

" <td>Mystery</td>\n",

" <td>29903244700.35</td>\n",

" <td>2.49</td>\n",

" </tr>\n",

" <tr>\n",

" <th>11</th>\n",

" <td>Horror</td>\n",

" <td>28800384751.85</td>\n",

" <td>2.39</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>Biography</td>\n",

" <td>16776660619.24</td>\n",

" <td>1.39</td>\n",

" </tr>\n",

" <tr>\n",

" <th>10</th>\n",

" <td>History</td>\n",

" <td>8429562660.69</td>\n",

" <td>0.70</td>\n",

" </tr>\n",

" <tr>\n",

" <th>12</th>\n",

" <td>Music</td>\n",

" <td>7439929226.68</td>\n",

" <td>0.62</td>\n",

" </tr>\n",

" <tr>\n",

" <th>13</th>\n",

" <td>Musical</td>\n",

" <td>5228065825.20</td>\n",

" <td>0.43</td>\n",

" </tr>\n",

" <tr>\n",

" <th>17</th>\n",

" <td>Sport</td>\n",

" <td>4620549486.84</td>\n",

" <td>0.38</td>\n",

" </tr>\n",

" <tr>\n",

" <th>19</th>\n",

" <td>War</td>\n",

" <td>4619522490.02</td>\n",

" <td>0.38</td>\n",

" </tr>\n",

" <tr>\n",

" <th>20</th>\n",

" <td>Western</td>\n",

" <td>2551516786.77</td>\n",

" <td>0.21</td>\n",

" </tr>\n",

" <tr>\n",

" <th>9</th>\n",

" <td>Film-Noir</td>\n",

" <td>153313504.88</td>\n",

" <td>0.01</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Genre Adjusted\_Profit Percent Total of Net Profit\n",

"1 Adventure 217335741708.40 18.07\n",

"0 Action 178930045524.32 14.88\n",

"4 Comedy 151922895671.69 12.63\n",

"6 Drama 144990041873.71 12.05\n",

"16 Sci-Fi 70465612908.78 5.86\n",

"2 Animation 68720987812.40 5.71\n",

"18 Thriller 65442236225.98 5.44\n",

"8 Fantasy 54797139085.80 4.56\n",

"15 Romance 50510744180.92 4.20\n",

"7 Family 46040638020.14 3.83\n",

"5 Crime 45194406614.69 3.76\n",

"14 Mystery 29903244700.35 2.49\n",

"11 Horror 28800384751.85 2.39\n",

"3 Biography 16776660619.24 1.39\n",

"10 History 8429562660.69 0.70\n",

"12 Music 7439929226.68 0.62\n",

"13 Musical 5228065825.20 0.43\n",

"17 Sport 4620549486.84 0.38\n",

"19 War 4619522490.02 0.38\n",

"20 Western 2551516786.77 0.21\n",

"9 Film-Noir 153313504.88 0.01"

]

},

"execution\_count": 39,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#Grouped by genre, find the percent total of the net profit for each.\n",

"per\_by\_genre = genre\_budgets\_df.groupby(['Genre'], as\_index=False)['Adjusted\_Profit'].sum().sort\_values(by='Adjusted\_Profit', ascending=False)\n",

"per\_by\_genre['Percent Total of Net Profit'] = (per\_by\_genre['Adjusted\_Profit']/per\_by\_genre['Adjusted\_Profit'].sum()\*100).round(2)\n",

"per\_by\_genre"

]

},

{

"cell\_type": "code",

"execution\_count": 40,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1008x504 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"#Plot the above findings.\n",

"plt.figure(figsize=(14,7))\n",

"ax6 = sns.barplot(x=per\_by\_genre['Percent Total of Net Profit'], y=per\_by\_genre['Genre'], palette='GnBu\_d')\n",

"plt.xlabel('Percent Total of Net Profit', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.ylabel('Genre', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('Percent of Net Profit By Genre', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.savefig('PercentProfitGenre');"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Now we can see that adventure, action, comedy and drama make up the lionshare of the overall net profits from all movies. However, from our recent observations we know there are also major opportunities in the animation and sci-fi markets due to lower saturation but high average net profits. We will soon determine which genres are most successful during which months.\n",

"\n",

"\*\*Question 2 Conclusion\*\*: We recommend that our Company should focus their efforts on the top 6 most profitable movie genres: Adventure, Action, Comedy, Drama, Sci-Fi and Animation. A further recommendation to focus on Sci-Fi and Animation due to less competition and a higher opportunity to profit."

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"# Question 3: What is the best time of the year to release a movie?"

]

},

{

"cell\_type": "code",

"execution\_count": 41,

"metadata": {},

"outputs": [],

"source": [

"#Convert the Release Date field to type datetime.\n",

"imdb\_budgets\_df['Release Date'] = pd.to\_datetime(imdb\_budgets\_df['Release Date'])"

]

},

{

"cell\_type": "code",

"execution\_count": 42,

"metadata": {},

"outputs": [],

"source": [

"#Add a new column called month, displaying only the month from the release date.\n",

"dateData = [x.strftime('%B') for x in imdb\_budgets\_df['Release Date']]\n",

"imdb\_budgets\_df['Month'] = dateData"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Let's first start by determing which months see the most movie releases."

]

},

{

"cell\_type": "code",

"execution\_count": 43,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Month</th>\n",

" <th>Movie</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>December</td>\n",

" <td>452</td>\n",

" </tr>\n",

" <tr>\n",

" <th>10</th>\n",

" <td>October</td>\n",

" <td>424</td>\n",

" </tr>\n",

" <tr>\n",

" <th>9</th>\n",

" <td>November</td>\n",

" <td>366</td>\n",

" </tr>\n",

" <tr>\n",

" <th>11</th>\n",

" <td>September</td>\n",

" <td>358</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>August</td>\n",

" <td>346</td>\n",

" </tr>\n",

" <tr>\n",

" <th>6</th>\n",

" <td>June</td>\n",

" <td>327</td>\n",

" </tr>\n",

" <tr>\n",

" <th>5</th>\n",

" <td>July</td>\n",

" <td>296</td>\n",

" </tr>\n",

" <tr>\n",

" <th>7</th>\n",

" <td>March</td>\n",

" <td>270</td>\n",

" </tr>\n",

" <tr>\n",

" <th>8</th>\n",

" <td>May</td>\n",

" <td>241</td>\n",

" </tr>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>April</td>\n",

" <td>238</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>February</td>\n",

" <td>236</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>January</td>\n",

" <td>186</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Month Movie\n",

"2 December 452\n",

"10 October 424\n",

"9 November 366\n",

"11 September 358\n",

"1 August 346\n",

"6 June 327\n",

"5 July 296\n",

"7 March 270\n",

"8 May 241\n",

"0 April 238\n",

"3 February 236\n",

"4 January 186"

]

},

"execution\_count": 43,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#Count the total number of movies and group by month.\n",

"m\_by\_month = imdb\_budgets\_df.groupby(['Month'], as\_index=False)['Movie'].count().sort\_values(by='Movie', ascending=False)\n",

"m\_by\_month"

]

},

{

"cell\_type": "code",

"execution\_count": 44,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1008x504 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"#Plot the above findings in order by month.\n",

"plt.figure(figsize=(14,7))\n",

"ax7 = sns.countplot(x=imdb\_budgets\_df['Month'], palette='Greens',\n",

" order=['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'])\n",

"plt.xlabel('Month', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.ylabel('Count', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('Count of Movie Release By Month', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.savefig('CountbyMonth', dpi=300);"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"As you can see Decemeber and October lead the way in terms of sheer quantity of movies but does this suggest a higher level of profitability? Next we will look into the average net income by movie for each month."

]

},

{

"cell\_type": "code",

"execution\_count": 45,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Month</th>\n",

" <th>Adjusted\_Profit</th>\n",

" <th>Profit\_Margin</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>6</th>\n",

" <td>June</td>\n",

" <td>80327640.00</td>\n",

" <td>0.64</td>\n",

" </tr>\n",

" <tr>\n",

" <th>8</th>\n",

" <td>May</td>\n",

" <td>79372161.65</td>\n",

" <td>0.64</td>\n",

" </tr>\n",

" <tr>\n",

" <th>5</th>\n",

" <td>July</td>\n",

" <td>74716618.14</td>\n",

" <td>0.64</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>December</td>\n",

" <td>56823086.46</td>\n",

" <td>0.59</td>\n",

" </tr>\n",

" <tr>\n",

" <th>9</th>\n",

" <td>November</td>\n",

" <td>47476647.51</td>\n",

" <td>0.57</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>February</td>\n",

" <td>41089454.38</td>\n",

" <td>0.56</td>\n",

" </tr>\n",

" <tr>\n",

" <th>7</th>\n",

" <td>March</td>\n",

" <td>33645813.78</td>\n",

" <td>0.51</td>\n",

" </tr>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>April</td>\n",

" <td>31435638.57</td>\n",

" <td>0.54</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>January</td>\n",

" <td>31132342.98</td>\n",

" <td>0.56</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>August</td>\n",

" <td>25383311.33</td>\n",

" <td>0.52</td>\n",

" </tr>\n",

" <tr>\n",

" <th>11</th>\n",

" <td>September</td>\n",

" <td>16430952.78</td>\n",

" <td>0.41</td>\n",

" </tr>\n",

" <tr>\n",

" <th>10</th>\n",

" <td>October</td>\n",

" <td>15579534.04</td>\n",

" <td>0.46</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Month Adjusted\_Profit Profit\_Margin\n",

"6 June 80327640.00 0.64\n",

"8 May 79372161.65 0.64\n",

"5 July 74716618.14 0.64\n",

"2 December 56823086.46 0.59\n",

"9 November 47476647.51 0.57\n",

"3 February 41089454.38 0.56\n",

"7 March 33645813.78 0.51\n",

"0 April 31435638.57 0.54\n",

"4 January 31132342.98 0.56\n",

"1 August 25383311.33 0.52\n",

"11 September 16430952.78 0.41\n",

"10 October 15579534.04 0.46"

]

},

"execution\_count": 45,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#Once again group the movies by month, showing the average net profit for each.\n",

"p\_by\_month = imdb\_budgets\_df.groupby('Month', as\_index=False)[['Adjusted\_Profit', 'Profit\_Margin']].median().sort\_values(by='Adjusted\_Profit', ascending=False)\n",

"p\_by\_month"

]

},

{

"cell\_type": "code",

"execution\_count": 46,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1008x504 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"#Plot your above findings in order by month.\n",

"plt.figure(figsize=(14,7))\n",

"ax8 = sns.barplot(x=p\_by\_month['Month'], y=p\_by\_month['Adjusted\_Profit'], palette='Greens', \n",

" order=['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'])\n",

"plt.xlabel('Month', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.ylabel('Net Profit (Tens of Million)', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('Net Profit By Release Month', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.savefig('ProfitbyMonth', dpi=300);"

]

},

{

"cell\_type": "code",

"execution\_count": 47,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1008x504 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"plt.figure(figsize=(14,7))\n",

"ax9 = sns.barplot(x=p\_by\_month['Month'], y=p\_by\_month['Profit\_Margin'], palette='Greens', \n",

" order=['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'])\n",

"plt.xlabel('Month', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.ylabel('Profit Margin', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('Profit Margin By Release Month', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.ylim(0.3, 0.7)\n",

"plt.savefig('MarginByMonth', dpi=300);"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Interestingly, May, June and July shoot to the top in terms of both median net profit and profit margin. It appears that the summer months tend to result in greater success, perhaps as a result of an influx of children and their parents during summer break. Now as previously mentioned, let's dig a little further and see which genre tends to do the best in which month."

]

},

{

"cell\_type": "code",

"execution\_count": 49,

"metadata": {},

"outputs": [],

"source": [

"#Convert the Release Date field to type datetime\n",

"#Add a new column called month, displaying only the month from the release date.\n",

"genre\_budgets\_df['Release Date'] = pd.to\_datetime(genre\_budgets\_df['Release Date'])\n",

"genreDate = [x.strftime('%B') for x in genre\_budgets\_df['Release Date']]\n",

"genre\_budgets\_df['Month'] = genreDate"

]

},

{

"cell\_type": "code",

"execution\_count": 50,

"metadata": {},

"outputs": [],

"source": [

"#Create a new table called month\_genre consisting of Genre, Month, Net Profit, and Release Date\n",

"month\_genre = genre\_budgets\_df[['Genre', 'Month', 'Adjusted\_Profit', 'Release Date']]\n",

"#Group by Genre and Month, displaying the average Net Profit for each combination.\n",

"month\_genre = month\_genre.groupby(['Genre', 'Month'], as\_index=False)['Adjusted\_Profit'].mean().sort\_values(by='Adjusted\_Profit', ascending=False)"

]

},

{

"cell\_type": "code",

"execution\_count": 51,

"metadata": {},

"outputs": [],

"source": [

"#Slice the top six most profitable genres from above.\n",

"Adventure\_df = month\_genre.loc[month\_genre['Genre'].str.contains('Adventure')]\n",

"Action\_df = month\_genre.loc[month\_genre['Genre'].str.contains('Action')]\n",

"Comedy\_df = month\_genre.loc[month\_genre['Genre'].str.contains('Comedy')]\n",

"Drama\_df = month\_genre.loc[month\_genre['Genre'].str.contains('Drama')]\n",

"Scifi\_df = month\_genre.loc[month\_genre['Genre'].str.contains('Sci-Fi')]\n",

"Animation\_df = month\_genre.loc[month\_genre['Genre'].str.contains('Animation')]"

]

},

{

"cell\_type": "code",

"execution\_count": 52,

"metadata": {},

"outputs": [],

"source": [

"#Concatenate the six new tables into one new table.\n",

"genre\_concat = [Adventure\_df, Action\_df, Comedy\_df, Drama\_df, Scifi\_df, Animation\_df]\n",

"month\_genre\_df = pd.concat(genre\_concat)"

]

},

{

"cell\_type": "code",

"execution\_count": 53,

"metadata": {},

"outputs": [],

"source": [

"#Create a table of the months in order.\n",

"months\_in\_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December']\n",

"#Create a pivot table of month\_genre\_df, use the month\_in\_order table to reindex the pivot table.\n",

"month\_genre\_pivoted = month\_genre\_df.pivot(index='Month', columns='Genre', values='Adjusted\_Profit').reindex(months\_in\_order)"

]

},

{

"cell\_type": "code",

"execution\_count": 54,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th>Genre</th>\n",

" <th>Action</th>\n",

" <th>Adventure</th>\n",

" <th>Animation</th>\n",

" <th>Comedy</th>\n",

" <th>Drama</th>\n",

" <th>Sci-Fi</th>\n",

" </tr>\n",

" <tr>\n",

" <th>Month</th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>January</th>\n",

" <td>67911226.86</td>\n",

" <td>101480251.68</td>\n",

" <td>254304586.21</td>\n",

" <td>70321717.64</td>\n",

" <td>43539017.01</td>\n",

" <td>86131136.28</td>\n",

" </tr>\n",

" <tr>\n",

" <th>February</th>\n",

" <td>107741220.58</td>\n",

" <td>149172991.22</td>\n",

" <td>273699863.40</td>\n",

" <td>78129901.96</td>\n",

" <td>63807537.49</td>\n",

" <td>123463145.04</td>\n",

" </tr>\n",

" <tr>\n",

" <th>March</th>\n",

" <td>127548996.11</td>\n",

" <td>200474749.59</td>\n",

" <td>240295152.35</td>\n",

" <td>81411129.63</td>\n",

" <td>52348133.09</td>\n",

" <td>171335731.24</td>\n",

" </tr>\n",

" <tr>\n",

" <th>April</th>\n",

" <td>259392394.58</td>\n",

" <td>371426341.09</td>\n",

" <td>135514583.52</td>\n",

" <td>50050513.61</td>\n",

" <td>77199294.63</td>\n",

" <td>400992743.36</td>\n",

" </tr>\n",

" <tr>\n",

" <th>May</th>\n",

" <td>300431780.23</td>\n",

" <td>376946029.72</td>\n",

" <td>587476204.76</td>\n",

" <td>187839907.64</td>\n",

" <td>96590740.22</td>\n",

" <td>381838680.03</td>\n",

" </tr>\n",

" <tr>\n",

" <th>June</th>\n",

" <td>265101499.32</td>\n",

" <td>392963586.66</td>\n",

" <td>587763663.68</td>\n",

" <td>175416615.42</td>\n",

" <td>112382070.55</td>\n",

" <td>366873462.47</td>\n",

" </tr>\n",

" <tr>\n",

" <th>July</th>\n",

" <td>257293527.76</td>\n",

" <td>280812330.30</td>\n",

" <td>325184250.83</td>\n",

" <td>140927144.14</td>\n",

" <td>119198995.62</td>\n",

" <td>262513716.23</td>\n",

" </tr>\n",

" <tr>\n",

" <th>August</th>\n",

" <td>78993517.46</td>\n",

" <td>81128041.19</td>\n",

" <td>108115881.94</td>\n",

" <td>52702618.10</td>\n",

" <td>65637106.34</td>\n",

" <td>80812011.13</td>\n",

" </tr>\n",

" <tr>\n",

" <th>September</th>\n",

" <td>52980175.19</td>\n",

" <td>93388465.69</td>\n",

" <td>153847514.52</td>\n",

" <td>53288686.20</td>\n",

" <td>40194497.00</td>\n",

" <td>110804792.63</td>\n",

" </tr>\n",

" <tr>\n",

" <th>October</th>\n",

" <td>124257794.43</td>\n",

" <td>169896169.96</td>\n",

" <td>249582645.96</td>\n",

" <td>46177500.88</td>\n",

" <td>42992650.53</td>\n",

" <td>100120506.83</td>\n",

" </tr>\n",

" <tr>\n",

" <th>November</th>\n",

" <td>139749410.88</td>\n",

" <td>303503861.24</td>\n",

" <td>354381890.29</td>\n",

" <td>165340406.04</td>\n",

" <td>87265604.53</td>\n",

" <td>210336333.85</td>\n",

" </tr>\n",

" <tr>\n",

" <th>December</th>\n",

" <td>235113158.91</td>\n",

" <td>295732977.48</td>\n",

" <td>202553251.30</td>\n",

" <td>128699177.32</td>\n",

" <td>117758948.19</td>\n",

" <td>113695722.89</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

"Genre Action Adventure Animation Comedy Drama \\\n",

"Month \n",

"January 67911226.86 101480251.68 254304586.21 70321717.64 43539017.01 \n",

"February 107741220.58 149172991.22 273699863.40 78129901.96 63807537.49 \n",

"March 127548996.11 200474749.59 240295152.35 81411129.63 52348133.09 \n",

"April 259392394.58 371426341.09 135514583.52 50050513.61 77199294.63 \n",

"May 300431780.23 376946029.72 587476204.76 187839907.64 96590740.22 \n",

"June 265101499.32 392963586.66 587763663.68 175416615.42 112382070.55 \n",

"July 257293527.76 280812330.30 325184250.83 140927144.14 119198995.62 \n",

"August 78993517.46 81128041.19 108115881.94 52702618.10 65637106.34 \n",

"September 52980175.19 93388465.69 153847514.52 53288686.20 40194497.00 \n",

"October 124257794.43 169896169.96 249582645.96 46177500.88 42992650.53 \n",

"November 139749410.88 303503861.24 354381890.29 165340406.04 87265604.53 \n",

"December 235113158.91 295732977.48 202553251.30 128699177.32 117758948.19 \n",

"\n",

"Genre Sci-Fi \n",

"Month \n",

"January 86131136.28 \n",

"February 123463145.04 \n",

"March 171335731.24 \n",

"April 400992743.36 \n",

"May 381838680.03 \n",

"June 366873462.47 \n",

"July 262513716.23 \n",

"August 80812011.13 \n",

"September 110804792.63 \n",

"October 100120506.83 \n",

"November 210336333.85 \n",

"December 113695722.89 "

]

},

"execution\_count": 54,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"month\_genre\_pivoted"

]

},

{

"cell\_type": "code",

"execution\_count": 63,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1584x720 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"#Visualize the top 6 most profitable genre's by month\n",

"ax10 = month\_genre\_pivoted.plot(kind='line', figsize=(22, 10), rot=0)\n",

"plt.legend(labelcolor='grey', loc='best', prop={'size': 15})\n",

"plt.xlabel('Month', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.ylabel('Net Profit(In Hundreds of Millions)', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('Net Profit by Month by Genre', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.xticks(fontsize=15, rotation=45)\n",

"plt.savefig('ProfitbyMonthbyGenre', dpi=300);"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"We can see that each genre follows the same basic pattern, with the summer months proving to be the most profitable time to release a movie. Some further analysis shows that releasing an animation movie in particular during the summer months will have the greatest potential for high net profits. On the other hand drama, although fluctuates slightly with the months, tends to have no impact based on release date. When considering what aspects go into creating a successful movie, it's clear that one must take take into account the impact of a well timed release date.\n",

"\n",

"\*\*Question 3 Conclusion\*\*: We recommend that our Company release the bulk of their movies, especially Animation, during the summer months. Adventure, Drama and Comedy movies would see similar success if released in November, but the recommendation remains to focus on summer."

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"# Question 4: Now that we've got a better understanding of what attributes to a successful movie, which actors and directors tend to add the most value?"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"In this section we are going to take a look at the average net profit across all movies. From there we want to determine which actors and directors consistently appear in movies where the net profit substantially exceeds the average. We will represent this in a field called Value Above Replacement(VAR). To further simplify this concept; if across all movies the average net profit is 100 dollars and the average net profit of movies from 'Actor: X' is 200 dollars he/she would have a VAR of 2. This number represents X times over the average. To eliminate outliers we will look at actors who appear in 10 or more movies and directors who work in 5 or more."

]

},

{

"cell\_type": "code",

"execution\_count": 64,

"metadata": {},

"outputs": [],

"source": [

"#Similar to the imdb\_budget\_df table let's start by adjusting for inflation.\n",

"actors\_df['Production Budget'] = (((2020-actors\_df['Year'])\*.0322)+1)\*actors\_df['Production Budget']\n",

"actors\_df['Worldwide Gross'] = (((2020-actors\_df['Year'])\*.0322)+1)\*actors\_df['Worldwide Gross']\n",

"actors\_df['Domestic Gross'] = (((2020-actors\_df['Year'])\*.0322)+1)\*actors\_df['Domestic Gross']"

]

},

{

"cell\_type": "code",

"execution\_count": 65,

"metadata": {},

"outputs": [],

"source": [

"#Calculate Net Profit and Profit Margin\n",

"actors\_df['Net Profit'] = actors\_df['Worldwide Gross'] - actors\_df['Production Budget']\n",

"actors\_df['Profit Margin'] = actors\_df['Net Profit'] / actors\_df['Worldwide Gross']"

]

},

{

"cell\_type": "code",

"execution\_count": 66,

"metadata": {},

"outputs": [],

"source": [

"#Let's filter the actors\_df table to only include actors that appeared in 10 or more movies\n",

"actor\_counts = actors\_df['value'].value\_counts()\n",

"actor\_list = actor\_counts[actor\_counts >= 10].index.tolist()\n",

"actors\_df = actors\_df[actors\_df['value'].isin(actor\_list)]"

]

},

{

"cell\_type": "code",

"execution\_count": 67,

"metadata": {},

"outputs": [],

"source": [

"#Calculate VAR, which is the average Net Profit by actor divided by average Net Profit for all movies.\n",

"actor\_total = actors\_df.groupby(['value'], as\_index=False)['Net Profit'].mean().sort\_values(by='Net Profit', ascending=False)\n",

"actor\_total['VAR'] = (actor\_total['Net Profit']/actor\_total['Net Profit'].mean())"

]

},

{

"cell\_type": "code",

"execution\_count": 68,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>value</th>\n",

" <th>Net Profit</th>\n",

" <th>VAR</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>113</th>\n",

" <td>Ian McKellen</td>\n",

" <td>642641141.05</td>\n",

" <td>4.56</td>\n",

" </tr>\n",

" <tr>\n",

" <th>88</th>\n",

" <td>Emma Watson</td>\n",

" <td>594070330.59</td>\n",

" <td>4.22</td>\n",

" </tr>\n",

" <tr>\n",

" <th>48</th>\n",

" <td>Chris Hemsworth</td>\n",

" <td>550993070.74</td>\n",

" <td>3.91</td>\n",

" </tr>\n",

" <tr>\n",

" <th>47</th>\n",

" <td>Chris Evans</td>\n",

" <td>518397913.83</td>\n",

" <td>3.68</td>\n",

" </tr>\n",

" <tr>\n",

" <th>262</th>\n",

" <td>Robert Downey Jr.</td>\n",

" <td>484884995.15</td>\n",

" <td>3.44</td>\n",

" </tr>\n",

" <tr>\n",

" <th>82</th>\n",

" <td>Elijah Wood</td>\n",

" <td>468414890.65</td>\n",

" <td>3.33</td>\n",

" </tr>\n",

" <tr>\n",

" <th>227</th>\n",

" <td>Mike Myers</td>\n",

" <td>451615981.41</td>\n",

" <td>3.21</td>\n",

" </tr>\n",

" <tr>\n",

" <th>324</th>\n",

" <td>Zoe Saldana</td>\n",

" <td>418413981.69</td>\n",

" <td>2.97</td>\n",

" </tr>\n",

" <tr>\n",

" <th>205</th>\n",

" <td>Mark Ruffalo</td>\n",

" <td>418051684.80</td>\n",

" <td>2.97</td>\n",

" </tr>\n",

" <tr>\n",

" <th>166</th>\n",

" <td>Josh Hutcherson</td>\n",

" <td>389946768.85</td>\n",

" <td>2.77</td>\n",

" </tr>\n",

" <tr>\n",

" <th>197</th>\n",

" <td>Leonardo DiCaprio</td>\n",

" <td>347929775.33</td>\n",

" <td>2.47</td>\n",

" </tr>\n",

" <tr>\n",

" <th>164</th>\n",

" <td>Josh Duhamel</td>\n",

" <td>347668686.44</td>\n",

" <td>2.47</td>\n",

" </tr>\n",

" <tr>\n",

" <th>178</th>\n",

" <td>Kathy Bates</td>\n",

" <td>347201332.37</td>\n",

" <td>2.47</td>\n",

" </tr>\n",

" <tr>\n",

" <th>316</th>\n",

" <td>Will Smith</td>\n",

" <td>336002549.53</td>\n",

" <td>2.39</td>\n",

" </tr>\n",

" <tr>\n",

" <th>138</th>\n",

" <td>Jennifer Lawrence</td>\n",

" <td>334744177.49</td>\n",

" <td>2.38</td>\n",

" </tr>\n",

" <tr>\n",

" <th>299</th>\n",

" <td>Tom Hanks</td>\n",

" <td>320791739.32</td>\n",

" <td>2.28</td>\n",

" </tr>\n",

" <tr>\n",

" <th>285</th>\n",

" <td>Shia LaBeouf</td>\n",

" <td>320522135.54</td>\n",

" <td>2.28</td>\n",

" </tr>\n",

" <tr>\n",

" <th>45</th>\n",

" <td>Chiwetel Ejiofor</td>\n",

" <td>311862722.21</td>\n",

" <td>2.21</td>\n",

" </tr>\n",

" <tr>\n",

" <th>308</th>\n",

" <td>Vin Diesel</td>\n",

" <td>309819051.08</td>\n",

" <td>2.20</td>\n",

" </tr>\n",

" <tr>\n",

" <th>78</th>\n",

" <td>Dwayne Johnson</td>\n",

" <td>308538514.10</td>\n",

" <td>2.19</td>\n",

" </tr>\n",

" <tr>\n",

" <th>108</th>\n",

" <td>Helena Bonham Carter</td>\n",

" <td>301712229.56</td>\n",

" <td>2.14</td>\n",

" </tr>\n",

" <tr>\n",

" <th>296</th>\n",

" <td>Tim Allen</td>\n",

" <td>298679367.49</td>\n",

" <td>2.12</td>\n",

" </tr>\n",

" <tr>\n",

" <th>36</th>\n",

" <td>Cameron Diaz</td>\n",

" <td>295720384.55</td>\n",

" <td>2.10</td>\n",

" </tr>\n",

" <tr>\n",

" <th>298</th>\n",

" <td>Tom Cruise</td>\n",

" <td>287290600.79</td>\n",

" <td>2.04</td>\n",

" </tr>\n",

" <tr>\n",

" <th>13</th>\n",

" <td>Anne Hathaway</td>\n",

" <td>286762937.70</td>\n",

" <td>2.04</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" value Net Profit VAR\n",

"113 Ian McKellen 642641141.05 4.56\n",

"88 Emma Watson 594070330.59 4.22\n",

"48 Chris Hemsworth 550993070.74 3.91\n",

"47 Chris Evans 518397913.83 3.68\n",

"262 Robert Downey Jr. 484884995.15 3.44\n",

"82 Elijah Wood 468414890.65 3.33\n",

"227 Mike Myers 451615981.41 3.21\n",

"324 Zoe Saldana 418413981.69 2.97\n",

"205 Mark Ruffalo 418051684.80 2.97\n",

"166 Josh Hutcherson 389946768.85 2.77\n",

"197 Leonardo DiCaprio 347929775.33 2.47\n",

"164 Josh Duhamel 347668686.44 2.47\n",

"178 Kathy Bates 347201332.37 2.47\n",

"316 Will Smith 336002549.53 2.39\n",

"138 Jennifer Lawrence 334744177.49 2.38\n",

"299 Tom Hanks 320791739.32 2.28\n",

"285 Shia LaBeouf 320522135.54 2.28\n",

"45 Chiwetel Ejiofor 311862722.21 2.21\n",

"308 Vin Diesel 309819051.08 2.20\n",

"78 Dwayne Johnson 308538514.10 2.19\n",

"108 Helena Bonham Carter 301712229.56 2.14\n",

"296 Tim Allen 298679367.49 2.12\n",

"36 Cameron Diaz 295720384.55 2.10\n",

"298 Tom Cruise 287290600.79 2.04\n",

"13 Anne Hathaway 286762937.70 2.04"

]

},

"execution\_count": 68,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#Create new table consisting of top 25 actors by VAR.\n",

"top\_actors = actor\_total.head(25)\n",

"top\_actors"

]

},

{

"cell\_type": "code",

"execution\_count": 70,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1008x504 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"#Plot above finding and label the average of 1 with a black line.\n",

"plt.figure(figsize=(14,7))\n",

"ax11 = sns.barplot(x=top\_actors['VAR'], y=top\_actors['value'])\n",

"plt.axvline(1, ls='--', color='black', linewidth=3)\n",

"plt.xlabel('VAR', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.ylabel('Actor', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('VAR By Actor Compared to Average', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.savefig('VARActor', dpi=300);"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Wow, from this list we can see that all of these actors consistently appear in very profitable movies; anywhere from two times the norm to four and a half times the norm. When casting a movie this is a good short-list from where to start making calls."

]

},

{

"cell\_type": "code",

"execution\_count": 71,

"metadata": {},

"outputs": [],

"source": [

"#Adjust directors table for inflation.\n",

"directors\_df['Production Budget'] = (((2020-directors\_df['Year'])\*.0322)+1)\*directors\_df['Production Budget']\n",

"directors\_df['Worldwide Gross'] = (((2020-directors\_df['Year'])\*.0322)+1)\*directors\_df['Worldwide Gross']\n",

"directors\_df['Domestic Gross'] = (((2020-directors\_df['Year'])\*.0322)+1)\*directors\_df['Domestic Gross']"

]

},

{

"cell\_type": "code",

"execution\_count": 72,

"metadata": {},

"outputs": [],

"source": [

"#Calucalte Net Profit and Profit Margin.\n",

"directors\_df['Net Profit'] = directors\_df['Worldwide Gross'] - directors\_df['Production Budget']\n",

"directors\_df['Profit Margin'] = directors\_df['Net Profit'] / directors\_df['Worldwide Gross']"

]

},

{

"cell\_type": "code",

"execution\_count": 73,

"metadata": {},

"outputs": [],

"source": [

"#Let's filter the actors\_df table to only include actors that appeared in 5 or more movies.\n",

"director\_counts = directors\_df['value'].value\_counts()\n",

"director\_list = director\_counts[director\_counts >= 5].index.tolist()\n",

"directors\_df = directors\_df[directors\_df['value'].isin(director\_list)]"

]

},

{

"cell\_type": "code",

"execution\_count": 74,

"metadata": {},

"outputs": [],

"source": [

"#Calculate VAR, which is the average Net Profit by director divided by average Net Profit for all movies.\n",

"director\_total = directors\_df.groupby(['value'], as\_index=False)['Net Profit'].mean().sort\_values(by='Net Profit', ascending=False)\n",

"director\_total['VAR'] = (director\_total['Net Profit']/actor\_total['Net Profit'].mean())"

]

},

{

"cell\_type": "code",

"execution\_count": 75,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>value</th>\n",

" <th>Net Profit</th>\n",

" <th>VAR</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>78</th>\n",

" <td>James Cameron</td>\n",

" <td>1244750157.55</td>\n",

" <td>8.84</td>\n",

" </tr>\n",

" <tr>\n",

" <th>11</th>\n",

" <td>Anthony Russo</td>\n",

" <td>1017389415.62</td>\n",

" <td>7.22</td>\n",

" </tr>\n",

" <tr>\n",

" <th>89</th>\n",

" <td>Joe Russo</td>\n",

" <td>1017389415.62</td>\n",

" <td>7.22</td>\n",

" </tr>\n",

" <tr>\n",

" <th>115</th>\n",

" <td>Lee Unkrich</td>\n",

" <td>912067911.25</td>\n",

" <td>6.48</td>\n",

" </tr>\n",

" <tr>\n",

" <th>148</th>\n",

" <td>Peter Jackson</td>\n",

" <td>821878024.53</td>\n",

" <td>5.84</td>\n",

" </tr>\n",

" <tr>\n",

" <th>50</th>\n",

" <td>David Yates</td>\n",

" <td>688135205.04</td>\n",

" <td>4.89</td>\n",

" </tr>\n",

" <tr>\n",

" <th>104</th>\n",

" <td>Jon Favreau</td>\n",

" <td>628704113.52</td>\n",

" <td>4.46</td>\n",

" </tr>\n",

" <tr>\n",

" <th>129</th>\n",

" <td>Michael Bay</td>\n",

" <td>588804626.49</td>\n",

" <td>4.18</td>\n",

" </tr>\n",

" <tr>\n",

" <th>96</th>\n",

" <td>John Lasseter</td>\n",

" <td>577254528.66</td>\n",

" <td>4.10</td>\n",

" </tr>\n",

" <tr>\n",

" <th>31</th>\n",

" <td>Christopher Nolan</td>\n",

" <td>576508914.30</td>\n",

" <td>4.09</td>\n",

" </tr>\n",

" <tr>\n",

" <th>194</th>\n",

" <td>Tom McGrath</td>\n",

" <td>558026757.25</td>\n",

" <td>3.96</td>\n",

" </tr>\n",

" <tr>\n",

" <th>27</th>\n",

" <td>Carlos Saldanha</td>\n",

" <td>542327603.19</td>\n",

" <td>3.85</td>\n",

" </tr>\n",

" <tr>\n",

" <th>34</th>\n",

" <td>Conrad Vernon</td>\n",

" <td>533554799.18</td>\n",

" <td>3.79</td>\n",

" </tr>\n",

" <tr>\n",

" <th>19</th>\n",

" <td>Brad Bird</td>\n",

" <td>522918604.82</td>\n",

" <td>3.71</td>\n",

" </tr>\n",

" <tr>\n",

" <th>82</th>\n",

" <td>James Wan</td>\n",

" <td>517843475.89</td>\n",

" <td>3.68</td>\n",

" </tr>\n",

" <tr>\n",

" <th>58</th>\n",

" <td>Eric Darnell</td>\n",

" <td>506570978.60</td>\n",

" <td>3.60</td>\n",

" </tr>\n",

" <tr>\n",

" <th>188</th>\n",

" <td>Steven Spielberg</td>\n",

" <td>490403244.69</td>\n",

" <td>3.48</td>\n",

" </tr>\n",

" <tr>\n",

" <th>200</th>\n",

" <td>Wilfred Jackson</td>\n",

" <td>473675805.64</td>\n",

" <td>3.36</td>\n",

" </tr>\n",

" <tr>\n",

" <th>160</th>\n",

" <td>Rob Minkoff</td>\n",

" <td>453631830.01</td>\n",

" <td>3.22</td>\n",

" </tr>\n",

" <tr>\n",

" <th>62</th>\n",

" <td>Francis Lawrence</td>\n",

" <td>439117499.61</td>\n",

" <td>3.12</td>\n",

" </tr>\n",

" <tr>\n",

" <th>29</th>\n",

" <td>Chris Columbus</td>\n",

" <td>434315443.48</td>\n",

" <td>3.08</td>\n",

" </tr>\n",

" <tr>\n",

" <th>171</th>\n",

" <td>Ron Clements</td>\n",

" <td>396185896.16</td>\n",

" <td>2.81</td>\n",

" </tr>\n",

" <tr>\n",

" <th>101</th>\n",

" <td>John Musker</td>\n",

" <td>396185896.16</td>\n",

" <td>2.81</td>\n",

" </tr>\n",

" <tr>\n",

" <th>169</th>\n",

" <td>Roland Emmerich</td>\n",

" <td>391218701.44</td>\n",

" <td>2.78</td>\n",

" </tr>\n",

" <tr>\n",

" <th>175</th>\n",

" <td>Sam Raimi</td>\n",

" <td>364101893.22</td>\n",

" <td>2.59</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" value Net Profit VAR\n",

"78 James Cameron 1244750157.55 8.84\n",

"11 Anthony Russo 1017389415.62 7.22\n",

"89 Joe Russo 1017389415.62 7.22\n",

"115 Lee Unkrich 912067911.25 6.48\n",

"148 Peter Jackson 821878024.53 5.84\n",

"50 David Yates 688135205.04 4.89\n",

"104 Jon Favreau 628704113.52 4.46\n",

"129 Michael Bay 588804626.49 4.18\n",

"96 John Lasseter 577254528.66 4.10\n",

"31 Christopher Nolan 576508914.30 4.09\n",

"194 Tom McGrath 558026757.25 3.96\n",

"27 Carlos Saldanha 542327603.19 3.85\n",

"34 Conrad Vernon 533554799.18 3.79\n",

"19 Brad Bird 522918604.82 3.71\n",

"82 James Wan 517843475.89 3.68\n",

"58 Eric Darnell 506570978.60 3.60\n",

"188 Steven Spielberg 490403244.69 3.48\n",

"200 Wilfred Jackson 473675805.64 3.36\n",

"160 Rob Minkoff 453631830.01 3.22\n",

"62 Francis Lawrence 439117499.61 3.12\n",

"29 Chris Columbus 434315443.48 3.08\n",

"171 Ron Clements 396185896.16 2.81\n",

"101 John Musker 396185896.16 2.81\n",

"169 Roland Emmerich 391218701.44 2.78\n",

"175 Sam Raimi 364101893.22 2.59"

]

},

"execution\_count": 75,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#Create new table consisting of top 25 directors by VAR.\n",

"top\_directors = director\_total.head(25)\n",

"top\_directors"

]

},

{

"cell\_type": "code",

"execution\_count": 76,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1008x504 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"#Plot above finding and label the average of 1 with a black line.\n",

"plt.figure(figsize=(14,7))\n",

"ax12 = sns.barplot(x=top\_directors['VAR'], y=top\_directors['value'])\n",

"plt.axvline(1, ls='--', color='black', linewidth=3)\n",

"plt.xlabel('VAR', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.ylabel('Director', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('VAR By Director Compared to Average', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.savefig('VARDirector', dpi=300);"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"It appears that the most significant value added comes from the directors chair. James Cameron movies on average make almost nine times the amount of the average movie, this emphasizes what great leadership represents on a set. If we wanted to further investigate which actors and directors make the most impact it would be important to determine which genre of movies they appear in most.\n",

"\n",

"\*\*Question 4 Conclusion\*\*: We recommend that our Comapany focus their cast and crew search to individuals who consistently score at least 1.0 on the VAR score. We can, with a high level of confidence, conclude that these individuals will elevate the overall production."

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"# Question 5: How much should you spend on a movie to win an Oscar?\n",

"\n",

"In order to answer this question we'll first need to join the `imdb\_budgets\_df` dataframe and the `awards\_df` dataframe. As there may be movies with duplicate titles, we set the indices of both dataframes to the movie name and year so that matching data is correctly joined."

]

},

{

"cell\_type": "code",

"execution\_count": 77,

"metadata": {},

"outputs": [],

"source": [

"imdb\_budgets\_df.set\_index(['Movie','Year'], inplace=True)\n",

"awards\_df.set\_index(['film\_name', 'film\_year'], inplace=True)"

]

},

{

"cell\_type": "code",

"execution\_count": 78,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th></th>\n",

" <th>IMDb</th>\n",

" <th>Rating</th>\n",

" <th>Runtime</th>\n",

" <th>Genre</th>\n",

" <th>Release Date</th>\n",

" <th>Production Budget</th>\n",

" <th>Domestic Gross</th>\n",

" <th>Worldwide Gross</th>\n",

" <th>Profit</th>\n",

" <th>Profit\_Margin</th>\n",

" <th>Adjusted\_Budget</th>\n",

" <th>Adjusted\_Profit</th>\n",

" <th>Month</th>\n",

" <th>awards\_won</th>\n",

" <th>awards\_nominated</th>\n",

" <th>win\_rate</th>\n",

" </tr>\n",

" <tr>\n",

" <th>Movie</th>\n",

" <th>Year</th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>Avatar</th>\n",

" <th>2009</th>\n",

" <td>7.80</td>\n",

" <td>PG-13</td>\n",

" <td>162</td>\n",

" <td>[Action, Adventure, Fantasy]</td>\n",

" <td>2009-12-17</td>\n",

" <td>237000000</td>\n",

" <td>760507625</td>\n",

" <td>2788701337</td>\n",

" <td>2551701337</td>\n",

" <td>0.92</td>\n",

" <td>320945400.00</td>\n",

" <td>3455513950.57</td>\n",

" <td>December</td>\n",

" <td>3</td>\n",

" <td>9</td>\n",

" <td>0.33</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Black Panther</th>\n",

" <th>2018</th>\n",

" <td>7.30</td>\n",

" <td>PG-13</td>\n",

" <td>134</td>\n",

" <td>[Action, Adventure, Sci-Fi]</td>\n",

" <td>2018-02-13</td>\n",

" <td>200000000</td>\n",

" <td>700059566</td>\n",

" <td>1346103376</td>\n",

" <td>1146103376</td>\n",

" <td>0.85</td>\n",

" <td>212880000.00</td>\n",

" <td>1219912433.41</td>\n",

" <td>February</td>\n",

" <td>3</td>\n",

" <td>7</td>\n",

" <td>0.43</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Titanic</th>\n",

" <th>1997</th>\n",

" <td>7.80</td>\n",

" <td>PG-13</td>\n",

" <td>194</td>\n",

" <td>[Drama, Romance]</td>\n",

" <td>1997-12-18</td>\n",

" <td>200000000</td>\n",

" <td>659363944</td>\n",

" <td>2208208395</td>\n",

" <td>2008208395</td>\n",

" <td>0.91</td>\n",

" <td>348120000.00</td>\n",

" <td>3495487532.34</td>\n",

" <td>December</td>\n",

" <td>11</td>\n",

" <td>14</td>\n",

" <td>0.79</td>\n",

" </tr>\n",

" <tr>\n",

" <th>The Dark Knight</th>\n",

" <th>2008</th>\n",

" <td>9.00</td>\n",

" <td>PG-13</td>\n",

" <td>152</td>\n",

" <td>[Action, Crime, Drama]</td>\n",

" <td>2008-07-11</td>\n",

" <td>185000000</td>\n",

" <td>533720947</td>\n",

" <td>1000742751</td>\n",

" <td>815742751</td>\n",

" <td>0.82</td>\n",

" <td>256484000.00</td>\n",

" <td>1130945749.99</td>\n",

" <td>July</td>\n",

" <td>2</td>\n",

" <td>8</td>\n",

" <td>0.25</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Toy Story 4</th>\n",

" <th>2019</th>\n",

" <td>7.80</td>\n",

" <td>G</td>\n",

" <td>100</td>\n",

" <td>[Animation, Adventure, Comedy]</td>\n",

" <td>2019-06-20</td>\n",

" <td>200000000</td>\n",

" <td>434038008</td>\n",

" <td>1073394813</td>\n",

" <td>873394813</td>\n",

" <td>0.81</td>\n",

" <td>206440000.00</td>\n",

" <td>901518125.98</td>\n",

" <td>June</td>\n",

" <td>1</td>\n",

" <td>2</td>\n",

" <td>0.50</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" IMDb Rating Runtime Genre \\\n",

"Movie Year \n",

"Avatar 2009 7.80 PG-13 162 [Action, Adventure, Fantasy] \n",

"Black Panther 2018 7.30 PG-13 134 [Action, Adventure, Sci-Fi] \n",

"Titanic 1997 7.80 PG-13 194 [Drama, Romance] \n",

"The Dark Knight 2008 9.00 PG-13 152 [Action, Crime, Drama] \n",

"Toy Story 4 2019 7.80 G 100 [Animation, Adventure, Comedy] \n",

"\n",

" Release Date Production Budget Domestic Gross \\\n",

"Movie Year \n",

"Avatar 2009 2009-12-17 237000000 760507625 \n",

"Black Panther 2018 2018-02-13 200000000 700059566 \n",

"Titanic 1997 1997-12-18 200000000 659363944 \n",

"The Dark Knight 2008 2008-07-11 185000000 533720947 \n",

"Toy Story 4 2019 2019-06-20 200000000 434038008 \n",

"\n",

" Worldwide Gross Profit Profit\_Margin \\\n",

"Movie Year \n",

"Avatar 2009 2788701337 2551701337 0.92 \n",

"Black Panther 2018 1346103376 1146103376 0.85 \n",

"Titanic 1997 2208208395 2008208395 0.91 \n",

"The Dark Knight 2008 1000742751 815742751 0.82 \n",

"Toy Story 4 2019 1073394813 873394813 0.81 \n",

"\n",

" Adjusted\_Budget Adjusted\_Profit Month awards\_won \\\n",

"Movie Year \n",

"Avatar 2009 320945400.00 3455513950.57 December 3 \n",

"Black Panther 2018 212880000.00 1219912433.41 February 3 \n",

"Titanic 1997 348120000.00 3495487532.34 December 11 \n",

"The Dark Knight 2008 256484000.00 1130945749.99 July 2 \n",

"Toy Story 4 2019 206440000.00 901518125.98 June 1 \n",

"\n",

" awards\_nominated win\_rate \n",

"Movie Year \n",

"Avatar 2009 9 0.33 \n",

"Black Panther 2018 7 0.43 \n",

"Titanic 1997 14 0.79 \n",

"The Dark Knight 2008 8 0.25 \n",

"Toy Story 4 2019 2 0.50 "

]

},

"execution\_count": 78,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"budgets\_and\_awards = imdb\_budgets\_df.join(awards\_df, how='inner', on=['Movie', 'Year'])\n",

"budgets\_and\_awards.head()"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"We've successfully joined the two dataframes. Let's filter the dataframe to include movies where the profit is greater than 0."

]

},

{

"cell\_type": "code",

"execution\_count": 79,

"metadata": {},

"outputs": [],

"source": [

"nominated\_movies\_df = budgets\_and\_awards.loc[budgets\_and\_awards['Profit'] > 0]"

]

},

{

"cell\_type": "code",

"execution\_count": 82,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1152x432 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"plt.figure(figsize=(16,6))\n",

"sns.boxplot(x='Adjusted\_Budget', data=nominated\_movies\_df, showfliers=False, color='powderblue')\n",

"sns.stripplot(x='Adjusted\_Budget', data=nominated\_movies\_df)\n",

"plt.ticklabel\_format(axis='x', style='sci', scilimits=(6,6))\n",

"plt.xticks(fontsize=12)\n",

"plt.xlabel('Movie Budgets Adjusted for Inflation (Millions of Dollars)', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'});\n",

"plt.title('Distribution of Movie Budgets for Profitable Oscar Nominated Movies', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.savefig('Oscar\_Nominated', dpi=300);"

]

},

{

"cell\_type": "code",

"execution\_count": 83,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"count 331.00\n",

"mean 66479336.13\n",

"std 72497186.73\n",

"min 212790.00\n",

"25% 15425660.00\n",

"50% 37816500.00\n",

"75% 93598000.00\n",

"max 348300000.00\n",

"Name: Adjusted\_Budget, dtype: float64"

]

},

"execution\_count": 83,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"nominated\_movies\_df['Adjusted\_Budget'].describe()"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"By looking at the distribution of movie budgets we see that the majority of data is clustered in an area below $100 million dollars.\n",

"\n",

"We need to take this a step further as the above distribution includes movies that were nominated and won awards as well as movies that did not win awards. In order to properly answer our question we must win an Oscar.\n",

"\n",

"We could filter by win rate and exclude those movies that did not win anything, however our data would still include movies that were nominated in a single category and won. This would skew the win rate as there would be several movies with a win rate of 100%. Let's take a look at the mean and median win rate to establish a threshold for award nominations."

]

},

{

"cell\_type": "code",

"execution\_count": 84,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"count 330.00\n",

"mean 0.45\n",

"std 0.28\n",

"min 0.00\n",

"25% 0.25\n",

"50% 0.39\n",

"75% 0.60\n",

"max 1.00\n",

"Name: win\_rate, dtype: float64"

]

},

"execution\_count": 84,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"nominated\_movies\_df['win\_rate'].describe()\n",

"#Let's be conservative for win rate and use the median win rate\n",

"#That means we would need to be nominated for at least 3 awards in order to win 1 award."

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"The mean win rate is 44.8% but as we mentioned is skewed by those movies with only 1 nomination. The median win rate is 39.2% which should be less skewed by the data and is a more conservative number. Using the median win rate of 39.2%, our movie would need to be nominated for at least 3 awards in order to get at least one win. 3 nominations will be the cutoff."

]

},

{

"cell\_type": "code",

"execution\_count": 85,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"263\n"

]

},

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1152x432 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"nominated\_over\_three = nominated\_movies\_df.loc[nominated\_movies\_df['awards\_nominated'] >= 3]\n",

"print(len(nominated\_over\_three))\n",

"plt.figure(figsize=(16,6))\n",

"sns.boxplot(x=nominated\_over\_three['Adjusted\_Budget'], showfliers=False, color='powderblue')\n",

"sns.stripplot(x='Adjusted\_Budget', data=nominated\_over\_three)\n",

"plt.ticklabel\_format(axis='x', style='sci', scilimits=(6,6))\n",

"plt.xticks(fontsize=12)\n",

"plt.xlabel('Movie Budgets Adjusted for Inflation (Millions of Dollars)', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('Distribution of Movie Budgets for Movies With At Least 3 Nominations', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.savefig('3\_Nominations', dpi=300);"

]

},

{

"cell\_type": "code",

"execution\_count": 86,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"count 263.00\n",

"mean 62404651.14\n",

"std 69126844.12\n",

"min 1224990.00\n",

"25% 15482900.00\n",

"50% 35465000.00\n",

"75% 78132000.00\n",

"max 348120000.00\n",

"Name: Adjusted\_Budget, dtype: float64"

]

},

"execution\_count": 86,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"nominated\_over\_three['Adjusted\_Budget'].describe()"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"It's important to note that the box plot of the `nominated\_over\_three` dataframe has shrunk! This means that our filter has decreased our interquartile range for the movie budget. Since this range is smaller there should be less variability in the middle of the data set. Since we have adjusted budgets that are extreme outliers, it is best to use the median as the primary measure of central tendency. The median adjusted budget for this data is \\\\$35,465,000. \n",

" \n",

"\*\*Question 5 Conclusion\*\*: Our Company should spend at least $35,465,000 in order to make an Oscar-winning movie.\n",

"\n",

"\*It is also worth noting that the 75th percentile of the adjusted budget for movies with at least three nominations is $78,132,000. This is close to our recommendation of a \\\\$82 million budget for a profitable movie with a profit margin of approximately 80%.\*"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"# Question 6: What impact, if any, does runtime and movie rating have on Net Profit, Profit Margin and IMDb rating?"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Let's first start by analyzing the ratings. We want to include only movies rated G, PG, PG-13 or R."

]

},

{

"cell\_type": "code",

"execution\_count": 87,

"metadata": {},

"outputs": [],

"source": [

"rating\_counts = imdb\_budgets\_df['Rating'].value\_counts()\n",

"rating\_list = rating\_counts[rating\_counts >= 50].index.tolist()\n",

"rating\_df = imdb\_budgets\_df[imdb\_budgets\_df['Rating'].isin(rating\_list)]"

]

},

{

"cell\_type": "code",

"execution\_count": 88,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Movie</th>\n",

" <th>Year</th>\n",

" <th>IMDb</th>\n",

" <th>Rating</th>\n",

" <th>Runtime</th>\n",

" <th>Genre</th>\n",

" <th>Release Date</th>\n",

" <th>Production Budget</th>\n",

" <th>Domestic Gross</th>\n",

" <th>Worldwide Gross</th>\n",

" <th>Profit</th>\n",

" <th>Profit\_Margin</th>\n",

" <th>Adjusted\_Budget</th>\n",

" <th>Adjusted\_Profit</th>\n",

" <th>Month</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>Avengers: Endgame</td>\n",

" <td>2019</td>\n",

" <td>8.40</td>\n",

" <td>PG-13</td>\n",

" <td>181</td>\n",

" <td>[Action, Adventure, Drama]</td>\n",

" <td>2019-04-23</td>\n",

" <td>400000000</td>\n",

" <td>858373000</td>\n",

" <td>2797800564</td>\n",

" <td>2397800564</td>\n",

" <td>0.86</td>\n",

" <td>412880000.00</td>\n",

" <td>2475009742.16</td>\n",

" <td>April</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>Avatar</td>\n",

" <td>2009</td>\n",

" <td>7.80</td>\n",

" <td>PG-13</td>\n",

" <td>162</td>\n",

" <td>[Action, Adventure, Fantasy]</td>\n",

" <td>2009-12-17</td>\n",

" <td>237000000</td>\n",

" <td>760507625</td>\n",

" <td>2788701337</td>\n",

" <td>2551701337</td>\n",

" <td>0.92</td>\n",

" <td>320945400.00</td>\n",

" <td>3455513950.57</td>\n",

" <td>December</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>Black Panther</td>\n",

" <td>2018</td>\n",

" <td>7.30</td>\n",

" <td>PG-13</td>\n",

" <td>134</td>\n",

" <td>[Action, Adventure, Sci-Fi]</td>\n",

" <td>2018-02-13</td>\n",

" <td>200000000</td>\n",

" <td>700059566</td>\n",

" <td>1346103376</td>\n",

" <td>1146103376</td>\n",

" <td>0.85</td>\n",

" <td>212880000.00</td>\n",

" <td>1219912433.41</td>\n",

" <td>February</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>Avengers: Infinity War</td>\n",

" <td>2018</td>\n",

" <td>8.40</td>\n",

" <td>PG-13</td>\n",

" <td>149</td>\n",

" <td>[Action, Adventure, Sci-Fi]</td>\n",

" <td>2018-04-25</td>\n",

" <td>300000000</td>\n",

" <td>678815482</td>\n",

" <td>2048359754</td>\n",

" <td>1748359754</td>\n",

" <td>0.85</td>\n",

" <td>319320000.00</td>\n",

" <td>1860954122.16</td>\n",

" <td>April</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>Titanic</td>\n",

" <td>1997</td>\n",

" <td>7.80</td>\n",

" <td>PG-13</td>\n",

" <td>194</td>\n",

" <td>[Drama, Romance]</td>\n",

" <td>1997-12-18</td>\n",

" <td>200000000</td>\n",

" <td>659363944</td>\n",

" <td>2208208395</td>\n",

" <td>2008208395</td>\n",

" <td>0.91</td>\n",

" <td>348120000.00</td>\n",

" <td>3495487532.34</td>\n",

" <td>December</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Movie Year IMDb Rating Runtime \\\n",

"0 Avengers: Endgame 2019 8.40 PG-13 181 \n",

"1 Avatar 2009 7.80 PG-13 162 \n",

"2 Black Panther 2018 7.30 PG-13 134 \n",

"3 Avengers: Infinity War 2018 8.40 PG-13 149 \n",

"4 Titanic 1997 7.80 PG-13 194 \n",

"\n",

" Genre Release Date Production Budget \\\n",

"0 [Action, Adventure, Drama] 2019-04-23 400000000 \n",

"1 [Action, Adventure, Fantasy] 2009-12-17 237000000 \n",

"2 [Action, Adventure, Sci-Fi] 2018-02-13 200000000 \n",

"3 [Action, Adventure, Sci-Fi] 2018-04-25 300000000 \n",

"4 [Drama, Romance] 1997-12-18 200000000 \n",

"\n",

" Domestic Gross Worldwide Gross Profit Profit\_Margin \\\n",

"0 858373000 2797800564 2397800564 0.86 \n",

"1 760507625 2788701337 2551701337 0.92 \n",

"2 700059566 1346103376 1146103376 0.85 \n",

"3 678815482 2048359754 1748359754 0.85 \n",

"4 659363944 2208208395 2008208395 0.91 \n",

"\n",

" Adjusted\_Budget Adjusted\_Profit Month \n",

"0 412880000.00 2475009742.16 April \n",

"1 320945400.00 3455513950.57 December \n",

"2 212880000.00 1219912433.41 February \n",

"3 319320000.00 1860954122.16 April \n",

"4 348120000.00 3495487532.34 December "

]

},

"execution\_count": 88,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"rating\_df = rating\_df.reset\_index()\n",

"rating\_df.head()"

]

},

{

"cell\_type": "code",

"execution\_count": 89,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Rating</th>\n",

" <th>Movie</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>R</td>\n",

" <td>1631</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>PG-13</td>\n",

" <td>1339</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>PG</td>\n",

" <td>590</td>\n",

" </tr>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>G</td>\n",

" <td>93</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Rating Movie\n",

"3 R 1631\n",

"2 PG-13 1339\n",

"1 PG 590\n",

"0 G 93"

]

},

"execution\_count": 89,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#Count the total number of movies and group by month.\n",

"rating\_count = rating\_df.groupby(['Rating'], as\_index=False)['Movie'].count().sort\_values(by='Movie', ascending=False)\n",

"rating\_count"

]

},

{

"cell\_type": "code",

"execution\_count": 90,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Rating</th>\n",

" <th>Adjusted\_Profit</th>\n",

" <th>Profit\_Margin</th>\n",

" <th>IMDb</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>G</td>\n",

" <td>154376810.04</td>\n",

" <td>0.76</td>\n",

" <td>7.10</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>PG</td>\n",

" <td>75404192.25</td>\n",

" <td>0.62</td>\n",

" <td>6.50</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>PG-13</td>\n",

" <td>49565772.61</td>\n",

" <td>0.55</td>\n",

" <td>6.30</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>R</td>\n",

" <td>20402474.98</td>\n",

" <td>0.51</td>\n",

" <td>6.60</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Rating Adjusted\_Profit Profit\_Margin IMDb\n",

"0 G 154376810.04 0.76 7.10\n",

"1 PG 75404192.25 0.62 6.50\n",

"2 PG-13 49565772.61 0.55 6.30\n",

"3 R 20402474.98 0.51 6.60"

]

},

"execution\_count": 90,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#Group by Rating let's determine which has the highest median net profit and profit margin.\n",

"rating\_df2 = rating\_df.groupby(['Rating'], as\_index=False)[['Adjusted\_Profit', 'Profit\_Margin', 'IMDb']].median().sort\_values(by='Adjusted\_Profit', ascending=False)\n",

"rating\_df2"

]

},

{

"cell\_type": "code",

"execution\_count": 91,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1008x504 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"# Plot your above findings\n",

"plt.figure(figsize=(14,7))\n",

"ax13 = sns.boxplot( y=rating\_df[\"Rating\"], x=rating\_df[\"Adjusted\_Profit\"], showfliers=False, palette='Blues')\n",

"plt.xlabel('Average Net Profit (Hundreds of Millions)', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.ylabel('Movie Ratings', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('Net Profit By Rating', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.savefig('ProfitbyRating', dpi=300);"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"As you can see, G and PG rated movies tend to perform best and account for the smallest market share. This, like the animation genre, is another opportunity to enter the market in a highly profitable arena with fewer competitors. It would be interesting to see a breakdown of total net profit by genre by rating to get a better idea of which rating and genres go best together."

]

},

{

"cell\_type": "code",

"execution\_count": 92,

"metadata": {},

"outputs": [],

"source": [

"# First drop the rating column from genre\_budgets\_df and genre from rating\_df\n",

"genre\_rating\_df = genre\_budgets\_df.drop(['Rating'], axis=1)\n",

"rating\_df = rating\_df.drop(['Genre'], axis=1)"

]

},

{

"cell\_type": "code",

"execution\_count": 93,

"metadata": {},

"outputs": [],

"source": [

"# Merge the genre\_rating\_df table and rating\_df table\n",

"genre\_rating\_df = pd.merge(genre\_rating\_df, rating\_df)"

]

},

{

"cell\_type": "code",

"execution\_count": 94,

"metadata": {},

"outputs": [],

"source": [

"#Slice the top six most profitable genres.\n",

"Adv\_df = genre\_rating\_df.loc[genre\_rating\_df['Genre'].str.contains('Adventure')]\n",

"Act\_df = genre\_rating\_df.loc[genre\_rating\_df['Genre'].str.contains('Action')]\n",

"Com\_df = genre\_rating\_df.loc[genre\_rating\_df['Genre'].str.contains('Comedy')]\n",

"Dra\_df = genre\_rating\_df.loc[genre\_rating\_df['Genre'].str.contains('Drama')]\n",

"Sci\_df = genre\_rating\_df.loc[genre\_rating\_df['Genre'].str.contains('Sci-Fi')]\n",

"Ani\_df = genre\_rating\_df.loc[genre\_rating\_df['Genre'].str.contains('Animation')]\n",

"\n",

"genre\_concat = [Adv\_df, Act\_df, Com\_df, Dra\_df, Sci\_df, Ani\_df]\n",

"genre\_rating = pd.concat(genre\_concat)"

]

},

{

"cell\_type": "code",

"execution\_count": 95,

"metadata": {},

"outputs": [],

"source": [

"# Create a pivot table from genre\_rating\n",

"gr\_df = genre\_rating.groupby(['Genre', 'Rating'], as\_index=False)['Adjusted\_Profit'].sum().sort\_values(by='Adjusted\_Profit', ascending=False)\n",

"gr\_pivoted = gr\_df.pivot(index='Genre', columns='Rating', values='Adjusted\_Profit')"

]

},

{

"cell\_type": "code",

"execution\_count": 96,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th>Rating</th>\n",

" <th>G</th>\n",

" <th>PG</th>\n",

" <th>PG-13</th>\n",

" <th>R</th>\n",

" </tr>\n",

" <tr>\n",

" <th>Genre</th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>Action</th>\n",

" <td>476713962.52</td>\n",

" <td>24806502581.61</td>\n",

" <td>118476527154.35</td>\n",

" <td>34527820240.94</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Adventure</th>\n",

" <td>17497561206.41</td>\n",

" <td>74656830471.14</td>\n",

" <td>114180501731.83</td>\n",

" <td>10663312187.82</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Animation</th>\n",

" <td>20451774875.23</td>\n",

" <td>46792514260.78</td>\n",

" <td>682637577.33</td>\n",

" <td>120368587.97</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Comedy</th>\n",

" <td>14989898831.46</td>\n",

" <td>61733858474.80</td>\n",

" <td>44722618139.99</td>\n",

" <td>30095649966.62</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Drama</th>\n",

" <td>6452247472.37</td>\n",

" <td>19785801203.02</td>\n",

" <td>64695667306.22</td>\n",

" <td>50557666303.54</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Sci-Fi</th>\n",

" <td>575199818.94</td>\n",

" <td>4693467863.02</td>\n",

" <td>54045363674.82</td>\n",

" <td>11072810424.26</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

"Rating G PG PG-13 R\n",

"Genre \n",

"Action 476713962.52 24806502581.61 118476527154.35 34527820240.94\n",

"Adventure 17497561206.41 74656830471.14 114180501731.83 10663312187.82\n",

"Animation 20451774875.23 46792514260.78 682637577.33 120368587.97\n",

"Comedy 14989898831.46 61733858474.80 44722618139.99 30095649966.62\n",

"Drama 6452247472.37 19785801203.02 64695667306.22 50557666303.54\n",

"Sci-Fi 575199818.94 4693467863.02 54045363674.82 11072810424.26"

]

},

"execution\_count": 96,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"# Preview the table.\n",

"gr\_pivoted"

]

},

{

"cell\_type": "code",

"execution\_count": 99,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1008x504 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"# Plot the above findings.\n",

"ax14 = gr\_pivoted.plot(kind='bar', stacked=True, figsize=(14,7))\n",

"plt.legend(labelcolor='grey', prop={'size': 16})\n",

"plt.xlabel('Genre', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.ylabel('Net Profit(Hundreds of Billions)', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('Net Profit By Genre By Rating', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.xticks(fontsize=15, rotation=45)\n",

"plt.savefig('ProfitbyGenrebyRating');"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"As one could have probably guessed, animation is almost entirely made up of G and PG rated movies. We can see that for most other genres, the bulk of their total net profits come from PG-13 rated movies. From this we can focus on which rating to aim for in each genre to evoke the most success.\n",

"\n",

"Now let's shift our focus to the film's runtime. Does movie length have an impact in terms of success?"

]

},

{

"cell\_type": "code",

"execution\_count": 100,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th></th>\n",

" <th>Runtime</th>\n",

" <th>Adjusted\_Profit</th>\n",

" <th>Profit\_Margin</th>\n",

" </tr>\n",

" <tr>\n",

" <th>Movie</th>\n",

" <th>Year</th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>Avengers: Endgame</th>\n",

" <th>2019</th>\n",

" <td>181</td>\n",

" <td>2475009742.16</td>\n",

" <td>0.86</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Avatar</th>\n",

" <th>2009</th>\n",

" <td>162</td>\n",

" <td>3455513950.57</td>\n",

" <td>0.92</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Black Panther</th>\n",

" <th>2018</th>\n",

" <td>134</td>\n",

" <td>1219912433.41</td>\n",

" <td>0.85</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Avengers: Infinity War</th>\n",

" <th>2018</th>\n",

" <td>149</td>\n",

" <td>1860954122.16</td>\n",

" <td>0.85</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Titanic</th>\n",

" <th>1997</th>\n",

" <td>194</td>\n",

" <td>3495487532.34</td>\n",

" <td>0.91</td>\n",

" </tr>\n",

" <tr>\n",

" <th>...</th>\n",

" <th>...</th>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>The Misfits</th>\n",

" <th>1961</th>\n",

" <td>125</td>\n",

" <td>12179160.00</td>\n",

" <td>0.51</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Judgment at Nuremberg</th>\n",

" <th>1961</th>\n",

" <td>179</td>\n",

" <td>20298600.00</td>\n",

" <td>0.70</td>\n",

" </tr>\n",

" <tr>\n",

" <th>The Wrong Man</th>\n",

" <th>1956</th>\n",

" <td>105</td>\n",

" <td>2448640.00</td>\n",

" <td>0.40</td>\n",

" </tr>\n",

" <tr>\n",

" <th>The Trouble with Harry</th>\n",

" <th>1955</th>\n",

" <td>99</td>\n",

" <td>17939400.00</td>\n",

" <td>0.83</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Niagara</th>\n",

" <th>1953</th>\n",

" <td>92</td>\n",

" <td>3946750.00</td>\n",

" <td>0.50</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"<p>3740 rows × 3 columns</p>\n",

"</div>"

],

"text/plain": [

" Runtime Adjusted\_Profit Profit\_Margin\n",

"Movie Year \n",

"Avengers: Endgame 2019 181 2475009742.16 0.86\n",

"Avatar 2009 162 3455513950.57 0.92\n",

"Black Panther 2018 134 1219912433.41 0.85\n",

"Avengers: Infinity War 2018 149 1860954122.16 0.85\n",

"Titanic 1997 194 3495487532.34 0.91\n",

"... ... ... ...\n",

"The Misfits 1961 125 12179160.00 0.51\n",

"Judgment at Nuremberg 1961 179 20298600.00 0.70\n",

"The Wrong Man 1956 105 2448640.00 0.40\n",

"The Trouble with Harry 1955 99 17939400.00 0.83\n",

"Niagara 1953 92 3946750.00 0.50\n",

"\n",

"[3740 rows x 3 columns]"

]

},

"execution\_count": 100,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"# Create a new table with runtime, net profit and profit margin.\n",

"runtime\_df = imdb\_budgets\_df[['Runtime', 'Adjusted\_Profit', 'Profit\_Margin']]\n",

"runtime\_df"

]

},

{

"cell\_type": "code",

"execution\_count": 101,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Runtime</th>\n",

" <th>Adjusted\_Profit</th>\n",

" <th>Profit\_Margin</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>Runtime</th>\n",

" <td>1.00</td>\n",

" <td>0.22</td>\n",

" <td>0.05</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Adjusted\_Profit</th>\n",

" <td>0.22</td>\n",

" <td>1.00</td>\n",

" <td>0.05</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Profit\_Margin</th>\n",

" <td>0.05</td>\n",

" <td>0.05</td>\n",

" <td>1.00</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Runtime Adjusted\_Profit Profit\_Margin\n",

"Runtime 1.00 0.22 0.05\n",

"Adjusted\_Profit 0.22 1.00 0.05\n",

"Profit\_Margin 0.05 0.05 1.00"

]

},

"execution\_count": 101,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"# Let's start by taking a look at the correlation between runtime and net profit/profit margin.\n",

"pearsoncorr = runtime\_df.corr(method='pearson')\n",

"pearsoncorr"

]

},

{

"cell\_type": "code",

"execution\_count": 102,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1008x504 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"# Plot the correlation.\n",

"plt.figure(figsize=(14,7))\n",

"ax15 = sns.regplot(x='Runtime', y='Adjusted\_Profit', data=imdb\_budgets\_df)\n",

"plt.xlabel('Movie Runtime', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.ylabel('Net Profit (In Billions)', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('Correlation Between Net Profit and Runtime', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.savefig('CorrProfitRuntime', dpi=300);"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Although there is a small positive correlation of .223 showing that the long the runtime the higher the net profit, it's incredibly minute. With that in mind, we can take from this that, typically, it is not important to keep a movie above or below a cetain time threshold.\n",

"\n",

"\*\*Question 6 Conclusion\*\*: We recommend that our Company take into consideration the rating of the movie based on the genre and target audience. If making animation movies, it is wise to stick to a G or PG rating, otherwise PG-13 is the sweetspot. In terms of runtime, there is little correlation in terms of overall profitability."

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"# Question 7: Sticking to our analysis of Net Profit and Profit Margin, what should our Company determine to be the baseline for sustainable success?"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"We have an understanding of what goes into a successful movie but let's determine what our Comapny should expect in terms of profitability if they expect to compete with the other top movie studios."

]

},

{

"cell\_type": "code",

"execution\_count": 103,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>title</th>\n",

" <th>studio</th>\n",

" <th>domestic\_gross</th>\n",

" <th>foreign\_gross</th>\n",

" <th>year</th>\n",

" <th>IMDb</th>\n",

" <th>Rating</th>\n",

" <th>Runtime</th>\n",

" <th>Genre</th>\n",

" <th>Release Date</th>\n",

" <th>Production Budget</th>\n",

" <th>Domestic Gross</th>\n",

" <th>Worldwide Gross</th>\n",

" <th>Profit</th>\n",

" <th>Profit\_Margin</th>\n",

" <th>Adjusted\_Budget</th>\n",

" <th>Adjusted\_Profit</th>\n",

" <th>Month</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>Toy Story 3</td>\n",

" <td>Buena Vista</td>\n",

" <td>415000000.00</td>\n",

" <td>652000000</td>\n",

" <td>2010</td>\n",

" <td>8.30</td>\n",

" <td>G</td>\n",

" <td>103</td>\n",

" <td>[Animation, Adventure, Comedy]</td>\n",

" <td>2010-06-18</td>\n",

" <td>200000000</td>\n",

" <td>415004880</td>\n",

" <td>1068879522</td>\n",

" <td>868879522</td>\n",

" <td>0.81</td>\n",

" <td>264400000.00</td>\n",

" <td>1148658728.08</td>\n",

" <td>June</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>Inception</td>\n",

" <td>WB</td>\n",

" <td>292600000.00</td>\n",

" <td>535700000</td>\n",

" <td>2010</td>\n",

" <td>8.80</td>\n",

" <td>PG-13</td>\n",

" <td>148</td>\n",

" <td>[Action, Adventure, Sci-Fi]</td>\n",

" <td>2010-07-16</td>\n",

" <td>160000000</td>\n",

" <td>292576195</td>\n",

" <td>832551961</td>\n",

" <td>672551961</td>\n",

" <td>0.81</td>\n",

" <td>211520000.00</td>\n",

" <td>889113692.44</td>\n",

" <td>July</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>Shrek Forever After</td>\n",

" <td>Pixar/Dreamworks</td>\n",

" <td>238700000.00</td>\n",

" <td>513900000</td>\n",

" <td>2010</td>\n",

" <td>6.30</td>\n",

" <td>PG</td>\n",

" <td>93</td>\n",

" <td>[Animation, Adventure, Comedy]</td>\n",

" <td>2010-05-21</td>\n",

" <td>165000000</td>\n",

" <td>238736787</td>\n",

" <td>756244673</td>\n",

" <td>591244673</td>\n",

" <td>0.78</td>\n",

" <td>218130000.00</td>\n",

" <td>781625457.71</td>\n",

" <td>May</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>The Twilight Saga: Eclipse</td>\n",

" <td>Sumbadhat Productions</td>\n",

" <td>300500000.00</td>\n",

" <td>398000000</td>\n",

" <td>2010</td>\n",

" <td>5.00</td>\n",

" <td>PG-13</td>\n",

" <td>124</td>\n",

" <td>[Adventure, Drama, Fantasy]</td>\n",

" <td>2010-06-30</td>\n",

" <td>68000000</td>\n",

" <td>300531751</td>\n",

" <td>706102828</td>\n",

" <td>638102828</td>\n",

" <td>0.90</td>\n",

" <td>89896000.00</td>\n",

" <td>843571938.62</td>\n",

" <td>June</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>Iron Man 2</td>\n",

" <td>Paramount</td>\n",

" <td>312400000.00</td>\n",

" <td>311500000</td>\n",

" <td>2010</td>\n",

" <td>7.00</td>\n",

" <td>PG-13</td>\n",

" <td>124</td>\n",

" <td>[Action, Adventure, Sci-Fi]</td>\n",

" <td>2010-05-07</td>\n",

" <td>170000000</td>\n",

" <td>312433331</td>\n",

" <td>621156389</td>\n",

" <td>451156389</td>\n",

" <td>0.73</td>\n",

" <td>224740000.00</td>\n",

" <td>596428746.26</td>\n",

" <td>May</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" title studio domestic\_gross \\\n",

"0 Toy Story 3 Buena Vista 415000000.00 \n",

"1 Inception WB 292600000.00 \n",

"2 Shrek Forever After Pixar/Dreamworks 238700000.00 \n",

"3 The Twilight Saga: Eclipse Sumbadhat Productions 300500000.00 \n",

"4 Iron Man 2 Paramount 312400000.00 \n",

"\n",

" foreign\_gross year IMDb Rating Runtime Genre \\\n",

"0 652000000 2010 8.30 G 103 [Animation, Adventure, Comedy] \n",

"1 535700000 2010 8.80 PG-13 148 [Action, Adventure, Sci-Fi] \n",

"2 513900000 2010 6.30 PG 93 [Animation, Adventure, Comedy] \n",

"3 398000000 2010 5.00 PG-13 124 [Adventure, Drama, Fantasy] \n",

"4 311500000 2010 7.00 PG-13 124 [Action, Adventure, Sci-Fi] \n",

"\n",

" Release Date Production Budget Domestic Gross Worldwide Gross Profit \\\n",

"0 2010-06-18 200000000 415004880 1068879522 868879522 \n",

"1 2010-07-16 160000000 292576195 832551961 672551961 \n",

"2 2010-05-21 165000000 238736787 756244673 591244673 \n",

"3 2010-06-30 68000000 300531751 706102828 638102828 \n",

"4 2010-05-07 170000000 312433331 621156389 451156389 \n",

"\n",

" Profit\_Margin Adjusted\_Budget Adjusted\_Profit Month \n",

"0 0.81 264400000.00 1148658728.08 June \n",

"1 0.81 211520000.00 889113692.44 July \n",

"2 0.78 218130000.00 781625457.71 May \n",

"3 0.90 89896000.00 843571938.62 June \n",

"4 0.73 224740000.00 596428746.26 May "

]

},

"execution\_count": 103,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"# Merge studio\_df and imdb\_budgets\_df\n",

"studiobudgets\_df = pd.merge(studio\_df, imdb\_budgets\_df, left\_on = 'title', right\_on='Movie') \n",

"studiobudgets\_df.head()"

]

},

{

"cell\_type": "code",

"execution\_count": 237,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Studio</th>\n",

" <th>IMDb</th>\n",

" <th>Rating</th>\n",

" <th>Runtime</th>\n",

" <th>Genre</th>\n",

" <th>Release Date</th>\n",

" <th>Profit\_Margin</th>\n",

" <th>Adjusted\_Budget</th>\n",

" <th>Adjusted\_Profit</th>\n",

" <th>Month</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>Buena Vista</td>\n",

" <td>8.3</td>\n",

" <td>G</td>\n",

" <td>103</td>\n",

" <td>[Animation, Adventure, Comedy]</td>\n",

" <td>2010-06-18</td>\n",

" <td>0.812888</td>\n",

" <td>264400000.0</td>\n",

" <td>1.148659e+09</td>\n",

" <td>June</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>WB</td>\n",

" <td>8.8</td>\n",

" <td>PG-13</td>\n",

" <td>148</td>\n",

" <td>[Action, Adventure, Sci-Fi]</td>\n",

" <td>2010-07-16</td>\n",

" <td>0.807820</td>\n",

" <td>211520000.0</td>\n",

" <td>8.891137e+08</td>\n",

" <td>July</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>Pixar/Dreamworks</td>\n",

" <td>6.3</td>\n",

" <td>PG</td>\n",

" <td>93</td>\n",

" <td>[Animation, Adventure, Comedy]</td>\n",

" <td>2010-05-21</td>\n",

" <td>0.781817</td>\n",

" <td>218130000.0</td>\n",

" <td>7.816255e+08</td>\n",

" <td>May</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>Sumbadhat Productions</td>\n",

" <td>5.0</td>\n",

" <td>PG-13</td>\n",

" <td>124</td>\n",

" <td>[Adventure, Drama, Fantasy]</td>\n",

" <td>2010-06-30</td>\n",

" <td>0.903697</td>\n",

" <td>89896000.0</td>\n",

" <td>8.435719e+08</td>\n",

" <td>June</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>Paramount</td>\n",

" <td>7.0</td>\n",

" <td>PG-13</td>\n",

" <td>124</td>\n",

" <td>[Action, Adventure, Sci-Fi]</td>\n",

" <td>2010-05-07</td>\n",

" <td>0.726317</td>\n",

" <td>224740000.0</td>\n",

" <td>5.964287e+08</td>\n",

" <td>May</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Studio IMDb Rating Runtime \\\n",

"0 Buena Vista 8.3 G 103 \n",

"1 WB 8.8 PG-13 148 \n",

"2 Pixar/Dreamworks 6.3 PG 93 \n",

"3 Sumbadhat Productions 5.0 PG-13 124 \n",

"4 Paramount 7.0 PG-13 124 \n",

"\n",

" Genre Release Date Profit\_Margin \\\n",

"0 [Animation, Adventure, Comedy] 2010-06-18 0.812888 \n",

"1 [Action, Adventure, Sci-Fi] 2010-07-16 0.807820 \n",

"2 [Animation, Adventure, Comedy] 2010-05-21 0.781817 \n",

"3 [Adventure, Drama, Fantasy] 2010-06-30 0.903697 \n",

"4 [Action, Adventure, Sci-Fi] 2010-05-07 0.726317 \n",

"\n",

" Adjusted\_Budget Adjusted\_Profit Month \n",

"0 264400000.0 1.148659e+09 June \n",

"1 211520000.0 8.891137e+08 July \n",

"2 218130000.0 7.816255e+08 May \n",

"3 89896000.0 8.435719e+08 June \n",

"4 224740000.0 5.964287e+08 May "

]

},

"execution\_count": 237,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"# Let's remove some unnecessary fields.\n",

"studiobudgets\_df.drop(columns = {'title', 'domestic\_gross', 'Domestic Gross', 'foreign\_gross', 'year', 'Production Budget', 'Worldwide Gross', 'Profit'}, inplace = True)\n",

"studiobudgets\_df.rename(columns = {'studio':'Studio','Worldwide Gross ':'Worldwide Gross' }, inplace = True) \n",

"studiobudgets\_df.head()"

]

},

{

"cell\_type": "code",

"execution\_count": 254,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Studio</th>\n",

" <th>IMDb</th>\n",

" <th>Runtime</th>\n",

" <th>Profit\_Margin</th>\n",

" <th>Adjusted\_Budget</th>\n",

" <th>Adjusted\_Profit</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>51</th>\n",

" <td>UTV</td>\n",

" <td>8.45</td>\n",

" <td>141.5</td>\n",

" <td>0.958798</td>\n",

" <td>33747300.0</td>\n",

" <td>6.921112e+08</td>\n",

" </tr>\n",

" <tr>\n",

" <th>37</th>\n",

" <td>Pixar/Dreamworks</td>\n",

" <td>6.70</td>\n",

" <td>94.0</td>\n",

" <td>0.716170</td>\n",

" <td>182352000.0</td>\n",

" <td>4.921191e+08</td>\n",

" </tr>\n",

" <tr>\n",

" <th>9</th>\n",

" <td>Buena Vista</td>\n",

" <td>7.10</td>\n",

" <td>117.0</td>\n",

" <td>0.667056</td>\n",

" <td>176565000.0</td>\n",

" <td>1.928538e+08</td>\n",

" </tr>\n",

" <tr>\n",

" <th>28</th>\n",

" <td>MBox</td>\n",

" <td>7.80</td>\n",

" <td>158.0</td>\n",

" <td>0.624019</td>\n",

" <td>116082000.0</td>\n",

" <td>1.926625e+08</td>\n",

" </tr>\n",

" <tr>\n",

" <th>48</th>\n",

" <td>Strand</td>\n",

" <td>6.50</td>\n",

" <td>112.0</td>\n",

" <td>0.741792</td>\n",

" <td>50796000.0</td>\n",

" <td>1.459292e+08</td>\n",

" </tr>\n",

" <tr>\n",

" <th>45</th>\n",

" <td>Sony</td>\n",

" <td>6.30</td>\n",

" <td>105.0</td>\n",

" <td>0.658692</td>\n",

" <td>65796000.0</td>\n",

" <td>1.296401e+08</td>\n",

" </tr>\n",

" <tr>\n",

" <th>35</th>\n",

" <td>Paramount</td>\n",

" <td>6.40</td>\n",

" <td>110.0</td>\n",

" <td>0.639187</td>\n",

" <td>53053600.0</td>\n",

" <td>1.270562e+08</td>\n",

" </tr>\n",

" <tr>\n",

" <th>20</th>\n",

" <td>Fox</td>\n",

" <td>6.35</td>\n",

" <td>106.0</td>\n",

" <td>0.644465</td>\n",

" <td>65785200.0</td>\n",

" <td>1.171804e+08</td>\n",

" </tr>\n",

" <tr>\n",

" <th>52</th>\n",

" <td>Universal</td>\n",

" <td>6.20</td>\n",

" <td>108.0</td>\n",

" <td>0.686945</td>\n",

" <td>47728000.0</td>\n",

" <td>1.081619e+08</td>\n",

" </tr>\n",

" <tr>\n",

" <th>54</th>\n",

" <td>WB</td>\n",

" <td>6.60</td>\n",

" <td>113.5</td>\n",

" <td>0.542261</td>\n",

" <td>66914000.0</td>\n",

" <td>8.010906e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>15</th>\n",

" <td>Eros International</td>\n",

" <td>7.10</td>\n",

" <td>160.0</td>\n",

" <td>0.836702</td>\n",

" <td>10170820.0</td>\n",

" <td>5.211316e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>55</th>\n",

" <td>Wein/Dimension</td>\n",

" <td>5.90</td>\n",

" <td>96.0</td>\n",

" <td>0.750298</td>\n",

" <td>27474000.0</td>\n",

" <td>5.093755e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>44</th>\n",

" <td>Screen Gems</td>\n",

" <td>5.80</td>\n",

" <td>103.0</td>\n",

" <td>0.698444</td>\n",

" <td>30121600.0</td>\n",

" <td>5.004866e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>27</th>\n",

" <td>Lionsgate/Summit Entertainment</td>\n",

" <td>6.55</td>\n",

" <td>110.0</td>\n",

" <td>0.606561</td>\n",

" <td>44584000.0</td>\n",

" <td>4.695959e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>32</th>\n",

" <td>Neon</td>\n",

" <td>7.50</td>\n",

" <td>119.0</td>\n",

" <td>0.795529</td>\n",

" <td>12062600.0</td>\n",

" <td>4.693164e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>46</th>\n",

" <td>Sony Pictures</td>\n",

" <td>6.70</td>\n",

" <td>112.0</td>\n",

" <td>0.664717</td>\n",

" <td>26847000.0</td>\n",

" <td>4.222879e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>25</th>\n",

" <td>Lionsgate</td>\n",

" <td>6.15</td>\n",

" <td>103.5</td>\n",

" <td>0.601290</td>\n",

" <td>32376500.0</td>\n",

" <td>3.662573e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>49</th>\n",

" <td>Sumbadhat Productions</td>\n",

" <td>6.60</td>\n",

" <td>100.0</td>\n",

" <td>0.446140</td>\n",

" <td>41273600.0</td>\n",

" <td>3.615531e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>19</th>\n",

" <td>Focus Features</td>\n",

" <td>6.90</td>\n",

" <td>108.0</td>\n",

" <td>0.484553</td>\n",

" <td>20121600.0</td>\n",

" <td>3.370892e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>56</th>\n",

" <td>Weinstein Company</td>\n",

" <td>7.20</td>\n",

" <td>106.5</td>\n",

" <td>0.694665</td>\n",

" <td>18381000.0</td>\n",

" <td>3.362412e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>43</th>\n",

" <td>STX</td>\n",

" <td>6.40</td>\n",

" <td>104.0</td>\n",

" <td>0.528697</td>\n",

" <td>32898000.0</td>\n",

" <td>3.331053e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>40</th>\n",

" <td>Relativity Media</td>\n",

" <td>6.25</td>\n",

" <td>105.5</td>\n",

" <td>0.506080</td>\n",

" <td>33053600.0</td>\n",

" <td>2.923035e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>10</th>\n",

" <td>CBS</td>\n",

" <td>6.60</td>\n",

" <td>102.0</td>\n",

" <td>0.591352</td>\n",

" <td>21926600.0</td>\n",

" <td>2.688925e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>50</th>\n",

" <td>The Orchard</td>\n",

" <td>7.90</td>\n",

" <td>101.0</td>\n",

" <td>0.891707</td>\n",

" <td>2822000.0</td>\n",

" <td>2.323702e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>47</th>\n",

" <td>Sony Pictures Classics</td>\n",

" <td>7.20</td>\n",

" <td>109.0</td>\n",

" <td>0.600112</td>\n",

" <td>8041120.0</td>\n",

" <td>1.587142e+07</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" Studio IMDb Runtime Profit\_Margin \\\n",

"51 UTV 8.45 141.5 0.958798 \n",

"37 Pixar/Dreamworks 6.70 94.0 0.716170 \n",

"9 Buena Vista 7.10 117.0 0.667056 \n",

"28 MBox 7.80 158.0 0.624019 \n",

"48 Strand 6.50 112.0 0.741792 \n",

"45 Sony 6.30 105.0 0.658692 \n",

"35 Paramount 6.40 110.0 0.639187 \n",

"20 Fox 6.35 106.0 0.644465 \n",

"52 Universal 6.20 108.0 0.686945 \n",

"54 WB 6.60 113.5 0.542261 \n",

"15 Eros International 7.10 160.0 0.836702 \n",

"55 Wein/Dimension 5.90 96.0 0.750298 \n",

"44 Screen Gems 5.80 103.0 0.698444 \n",

"27 Lionsgate/Summit Entertainment 6.55 110.0 0.606561 \n",

"32 Neon 7.50 119.0 0.795529 \n",

"46 Sony Pictures 6.70 112.0 0.664717 \n",

"25 Lionsgate 6.15 103.5 0.601290 \n",

"49 Sumbadhat Productions 6.60 100.0 0.446140 \n",

"19 Focus Features 6.90 108.0 0.484553 \n",

"56 Weinstein Company 7.20 106.5 0.694665 \n",

"43 STX 6.40 104.0 0.528697 \n",

"40 Relativity Media 6.25 105.5 0.506080 \n",

"10 CBS 6.60 102.0 0.591352 \n",

"50 The Orchard 7.90 101.0 0.891707 \n",

"47 Sony Pictures Classics 7.20 109.0 0.600112 \n",

"\n",

" Adjusted\_Budget Adjusted\_Profit \n",

"51 33747300.0 6.921112e+08 \n",

"37 182352000.0 4.921191e+08 \n",

"9 176565000.0 1.928538e+08 \n",

"28 116082000.0 1.926625e+08 \n",

"48 50796000.0 1.459292e+08 \n",

"45 65796000.0 1.296401e+08 \n",

"35 53053600.0 1.270562e+08 \n",

"20 65785200.0 1.171804e+08 \n",

"52 47728000.0 1.081619e+08 \n",

"54 66914000.0 8.010906e+07 \n",

"15 10170820.0 5.211316e+07 \n",

"55 27474000.0 5.093755e+07 \n",

"44 30121600.0 5.004866e+07 \n",

"27 44584000.0 4.695959e+07 \n",

"32 12062600.0 4.693164e+07 \n",

"46 26847000.0 4.222879e+07 \n",

"25 32376500.0 3.662573e+07 \n",

"49 41273600.0 3.615531e+07 \n",

"19 20121600.0 3.370892e+07 \n",

"56 18381000.0 3.362412e+07 \n",

"43 32898000.0 3.331053e+07 \n",

"40 33053600.0 2.923035e+07 \n",

"10 21926600.0 2.688925e+07 \n",

"50 2822000.0 2.323702e+07 \n",

"47 8041120.0 1.587142e+07 "

]

},

"execution\_count": 254,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"# Group by studio, find median and filter to top 25 by Adjusted Profit\n",

"profit\_by\_studiodf = studiobudgets\_df.groupby('Studio').median()\n",

"profit\_by\_studiodf = profit\_by\_studiodf.reset\_index() \n",

"profit\_by\_studiodf = profit\_by\_studiodf.nlargest(25, 'Adjusted\_Profit')\n",

"profit\_by\_studiodf"

]

},

{

"cell\_type": "code",

"execution\_count": 253,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>IMDb</th>\n",

" <th>Runtime</th>\n",

" <th>Profit\_Margin</th>\n",

" <th>Adjusted\_Budget</th>\n",

" <th>Adjusted\_Profit</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>count</th>\n",

" <td>25.00000</td>\n",

" <td>25.000000</td>\n",

" <td>25.000000</td>\n",

" <td>2.500000e+01</td>\n",

" <td>2.500000e+01</td>\n",

" </tr>\n",

" <tr>\n",

" <th>mean</th>\n",

" <td>6.76600</td>\n",

" <td>112.180000</td>\n",

" <td>0.663049</td>\n",

" <td>4.883893e+07</td>\n",

" <td>1.134278e+08</td>\n",

" </tr>\n",

" <tr>\n",

" <th>std</th>\n",

" <td>0.64108</td>\n",

" <td>16.751169</td>\n",

" <td>0.122761</td>\n",

" <td>4.612474e+07</td>\n",

" <td>1.557719e+08</td>\n",

" </tr>\n",

" <tr>\n",

" <th>min</th>\n",

" <td>5.80000</td>\n",

" <td>94.000000</td>\n",

" <td>0.446140</td>\n",

" <td>2.822000e+06</td>\n",

" <td>1.587142e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>25%</th>\n",

" <td>6.35000</td>\n",

" <td>103.500000</td>\n",

" <td>0.600112</td>\n",

" <td>2.192660e+07</td>\n",

" <td>3.370892e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>50%</th>\n",

" <td>6.60000</td>\n",

" <td>108.000000</td>\n",

" <td>0.658692</td>\n",

" <td>3.305360e+07</td>\n",

" <td>5.004866e+07</td>\n",

" </tr>\n",

" <tr>\n",

" <th>75%</th>\n",

" <td>7.10000</td>\n",

" <td>112.000000</td>\n",

" <td>0.716170</td>\n",

" <td>5.305360e+07</td>\n",

" <td>1.270562e+08</td>\n",

" </tr>\n",

" <tr>\n",

" <th>max</th>\n",

" <td>8.45000</td>\n",

" <td>160.000000</td>\n",

" <td>0.958798</td>\n",

" <td>1.823520e+08</td>\n",

" <td>6.921112e+08</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" IMDb Runtime Profit\_Margin Adjusted\_Budget Adjusted\_Profit\n",

"count 25.00000 25.000000 25.000000 2.500000e+01 2.500000e+01\n",

"mean 6.76600 112.180000 0.663049 4.883893e+07 1.134278e+08\n",

"std 0.64108 16.751169 0.122761 4.612474e+07 1.557719e+08\n",

"min 5.80000 94.000000 0.446140 2.822000e+06 1.587142e+07\n",

"25% 6.35000 103.500000 0.600112 2.192660e+07 3.370892e+07\n",

"50% 6.60000 108.000000 0.658692 3.305360e+07 5.004866e+07\n",

"75% 7.10000 112.000000 0.716170 5.305360e+07 1.270562e+08\n",

"max 8.45000 160.000000 0.958798 1.823520e+08 6.921112e+08"

]

},

"execution\_count": 253,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"# Let's take a look at the average of these median values.\n",

"profit\_by\_studiodf.describe()"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"We can see that if we want to strive to be in the top half of this elite list of movie studios we need to have a profit margin of 66% and a net profit of 50 million per movie."

]

},

{

"cell\_type": "code",

"execution\_count": 256,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1008x504 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"#Plot the above findings.\n",

"plt.figure(figsize=(14,7))\n",

"ax16 = sns.barplot(x=profit\_by\_studiodf['Profit\_Margin'], y=profit\_by\_studiodf['Studio'],\n",

" order=profit\_by\_studiodf.sort\_values('Profit\_Margin', ascending=False).Studio, palette='GnBu\_d')\n",

"plt.xlabel('Profit Margin', fontsize=12)\n",

"plt.ylabel('Studio', fontsize=12)\n",

"plt.title('Median Profit Margin By Studio', fontsize=14)\n",

"plt.xlim(0.3, 1.0)\n",

"plt.savefig('ProfitMarginStudio')"

]

},

{

"cell\_type": "code",

"execution\_count": 255,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1008x504 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"#Plot the above findings.\n",

"plt.figure(figsize=(14,7))\n",

"ax16 = sns.barplot(y=profit\_by\_studiodf['Studio'], x=profit\_by\_studiodf['Adjusted\_Profit'], palette='GnBu\_d')\n",

"plt.xlabel('Net Profit (Hundreds of Millions)', fontsize=12)\n",

"plt.ylabel('Studio', fontsize=12)\n",

"plt.title('Median Net Profit By Studio', fontsize=14)\n",

"plt.savefig('NetProfitStudio');"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"We can see from the graph above that the major players in the studio industry have profit margins ranging from 24% to 95%. That's quite a large range to define success. However, the top 25 studios shown are many of the studios that we often recognize when we go to the movies. As we've done previously, we use the median profit margin of the top 25 as a target for success among major studios. That profit margin is 66%.\n",

"In the next analysis we'll take a closer look at some of these major studios to see what metrics we should try to mimic. Let's also keep this in mind as we go into our next analysis: UTV which has the greatest profit margin of all the studios is a subsidiary of Disney.\n",

"\n",

"\*\*Question 7 Conclusion\*\*: Microsoft should aim for a profit margin of 66% and a net profit of slightly over 50 million per movie to compete with the top existing studios."

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"# Question 8: Based on the success of current competitors, which should we look to for best practices?\n",

"\n",

"We need to add a column to the `theaters\_df` dataframe to calculate the money grossed per theater for a given movie. Then we can group by studio."

]

},

{

"cell\_type": "code",

"execution\_count": 106,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>title</th>\n",

" <th>max\_theaters</th>\n",

" <th>year</th>\n",

" <th>total\_dom\_gross($)</th>\n",

" <th>studio</th>\n",

" <th>dollars\_per\_theater</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>The Lion King</td>\n",

" <td>4802</td>\n",

" <td>2019</td>\n",

" <td>543638043</td>\n",

" <td>Disney</td>\n",

" <td>113210.75</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>Avengers: Endgame</td>\n",

" <td>4662</td>\n",

" <td>2019</td>\n",

" <td>858373000</td>\n",

" <td>Disney</td>\n",

" <td>184121.19</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>Spider-Man: Far from Home</td>\n",

" <td>4634</td>\n",

" <td>2019</td>\n",

" <td>390532085</td>\n",

" <td>Sony</td>\n",

" <td>84275.37</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>Toy Story 4</td>\n",

" <td>4575</td>\n",

" <td>2019</td>\n",

" <td>434038008</td>\n",

" <td>Disney</td>\n",

" <td>94871.70</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>It Chapter Two</td>\n",

" <td>4570</td>\n",

" <td>2019</td>\n",

" <td>211593228</td>\n",

" <td>Warner Bros.</td>\n",

" <td>46300.49</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" title max\_theaters year total\_dom\_gross($) \\\n",

"0 The Lion King 4802 2019 543638043 \n",

"1 Avengers: Endgame 4662 2019 858373000 \n",

"2 Spider-Man: Far from Home 4634 2019 390532085 \n",

"3 Toy Story 4 4575 2019 434038008 \n",

"4 It Chapter Two 4570 2019 211593228 \n",

"\n",

" studio dollars\_per\_theater \n",

"0 Disney 113210.75 \n",

"1 Disney 184121.19 \n",

"2 Sony 84275.37 \n",

"3 Disney 94871.70 \n",

"4 Warner Bros. 46300.49 "

]

},

"execution\_count": 106,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"theaters\_df['dollars\_per\_theater'] = theaters\_df['total\_dom\_gross($)'] / theaters\_df['max\_theaters']\n",

"theaters\_df.head()"

]

},

{

"cell\_type": "code",

"execution\_count": 107,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>max\_theaters</th>\n",

" <th>year</th>\n",

" <th>total\_dom\_gross($)</th>\n",

" <th>dollars\_per\_theater</th>\n",

" </tr>\n",

" <tr>\n",

" <th>studio</th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>Disney</th>\n",

" <td>3682.32</td>\n",

" <td>2010.59</td>\n",

" <td>202617891.97</td>\n",

" <td>51856.14</td>\n",

" </tr>\n",

" <tr>\n",

" <th>DreamWorks Distribution</th>\n",

" <td>3408.26</td>\n",

" <td>2002.95</td>\n",

" <td>118198315.42</td>\n",

" <td>33102.06</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Lionsgate</th>\n",

" <td>3356.24</td>\n",

" <td>2014.47</td>\n",

" <td>95268293.14</td>\n",

" <td>26485.34</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Metro-Goldwyn-Mayer (MGM)</th>\n",

" <td>3259.14</td>\n",

" <td>2004.00</td>\n",

" <td>78437576.64</td>\n",

" <td>23829.21</td>\n",

" </tr>\n",

" <tr>\n",

" <th>New Line Cinema</th>\n",

" <td>3410.57</td>\n",

" <td>2001.86</td>\n",

" <td>249718149.29</td>\n",

" <td>72518.24</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Paramount Pictures</th>\n",

" <td>3466.71</td>\n",

" <td>2010.71</td>\n",

" <td>108614912.30</td>\n",

" <td>30508.47</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Sony</th>\n",

" <td>3478.36</td>\n",

" <td>2010.56</td>\n",

" <td>116677932.63</td>\n",

" <td>32626.67</td>\n",

" </tr>\n",

" <tr>\n",

" <th>TriStar Pictures</th>\n",

" <td>3146.00</td>\n",

" <td>2014.00</td>\n",

" <td>80703217.29</td>\n",

" <td>25546.75</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Twentieth Century Fox</th>\n",

" <td>3493.98</td>\n",

" <td>2011.21</td>\n",

" <td>111009777.12</td>\n",

" <td>31119.14</td>\n",

" </tr>\n",

" <tr>\n",

" <th>United Artists</th>\n",

" <td>3124.00</td>\n",

" <td>2003.00</td>\n",

" <td>35667218.00</td>\n",

" <td>11417.16</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Universal Pictures</th>\n",

" <td>3488.41</td>\n",

" <td>2011.96</td>\n",

" <td>124914179.39</td>\n",

" <td>34679.48</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Warner Bros.</th>\n",

" <td>3535.03</td>\n",

" <td>2011.59</td>\n",

" <td>120355240.25</td>\n",

" <td>32678.01</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" max\_theaters year total\_dom\_gross($) \\\n",

"studio \n",

"Disney 3682.32 2010.59 202617891.97 \n",

"DreamWorks Distribution 3408.26 2002.95 118198315.42 \n",

"Lionsgate 3356.24 2014.47 95268293.14 \n",

"Metro-Goldwyn-Mayer (MGM) 3259.14 2004.00 78437576.64 \n",

"New Line Cinema 3410.57 2001.86 249718149.29 \n",

"Paramount Pictures 3466.71 2010.71 108614912.30 \n",

"Sony 3478.36 2010.56 116677932.63 \n",

"TriStar Pictures 3146.00 2014.00 80703217.29 \n",

"Twentieth Century Fox 3493.98 2011.21 111009777.12 \n",

"United Artists 3124.00 2003.00 35667218.00 \n",

"Universal Pictures 3488.41 2011.96 124914179.39 \n",

"Warner Bros. 3535.03 2011.59 120355240.25 \n",

"\n",

" dollars\_per\_theater \n",

"studio \n",

"Disney 51856.14 \n",

"DreamWorks Distribution 33102.06 \n",

"Lionsgate 26485.34 \n",

"Metro-Goldwyn-Mayer (MGM) 23829.21 \n",

"New Line Cinema 72518.24 \n",

"Paramount Pictures 30508.47 \n",

"Sony 32626.67 \n",

"TriStar Pictures 25546.75 \n",

"Twentieth Century Fox 31119.14 \n",

"United Artists 11417.16 \n",

"Universal Pictures 34679.48 \n",

"Warner Bros. 32678.01 "

]

},

"execution\_count": 107,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#Let's see what the average is for max number of theaters and for gross per theater for each studio\n",

"average\_theaters = theaters\_df.groupby('studio').mean()\n",

"average\_theaters\_ranked = average\_theaters.sort\_values(by=['studio'],ascending=False)\n",

"average\_theaters\_ranked.reset\_index(inplace=True)\n",

"average\_theaters"

]

},

{

"cell\_type": "code",

"execution\_count": 115,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1080x936 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"plt.figure(figsize=(15,13))\n",

"ax16 = sns.barplot(x='studio', y='dollars\_per\_theater', data=average\_theaters\_ranked)\n",

"plt.xlabel('Studio', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title(\"Average Domestic Gross Per Theater\", fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'});\n",

"plt.ylabel('Average Dollars Per Theater', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'});\n",

"plt.xticks(rotation=35, horizontalalignment='right')\n",

"plt.savefig('DomesticPerTheater', dpi=300);"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"In the bar plot above, Disney and New Line Cinema stand out. We need to double check that there are an appropriate number of movies by each of these studios before jumping to conclusions."

]

},

{

"cell\_type": "code",

"execution\_count": 116,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"Warner Bros. 208\n",

"Twentieth Century Fox 165\n",

"Disney 147\n",

"Universal Pictures 136\n",

"Sony 135\n",

"Paramount Pictures 112\n",

"Lionsgate 49\n",

"DreamWorks Distribution 19\n",

"Metro-Goldwyn-Mayer (MGM) 14\n",

"New Line Cinema 7\n",

"TriStar Pictures 7\n",

"United Artists 1\n",

"Name: studio, dtype: int64"

]

},

"execution\_count": 116,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"theaters\_df['studio'].value\_counts()"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"We can see that New Line Cinema only has 7 movies in this dataframe which means that their average domestic gross per theater is going to be skewed. Disney is certainly still a possibility and we should also consider Warner Bros. and Twentieth Century Fox."

]

},

{

"cell\_type": "code",

"execution\_count": 118,

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 1152x576 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"ax17 = sns.lmplot(x='max\_theaters', y='total\_dom\_gross($)', data=average\_theaters, height=8, aspect=2)\n",

"plt.ticklabel\_format(axis='y', style='sci', scilimits=(6,6))\n",

"plt.xlabel('Average of Maximum Theater Showings by Major Studio', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.ylabel('Average Domestic Gross (Millions of Dollars)', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '15'})\n",

"plt.title('Maximum Number of Theaters vs. Domestic Gross', fontdict = {'fontname': 'Times New Roman', 'color': 'gray', 'fontsize' : '25'})\n",

"plt.savefig('TheatersVGross', dpi=300);"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"The scatter plot shows a positive trend between the average number of theaters and the average domestic gross. The sole outlier is New Line Cinemas due to how few movies they are associated with in our dataframe. Disney is farthest to the right and above the trend line further proving that they should be a strong consideration.\n",

"\n",

"We'll join the theater and awards dataframes so that we can see which studios have the best win rate at the Oscars."

]

},

{

"cell\_type": "code",

"execution\_count": 119,

"metadata": {},

"outputs": [],

"source": [

"theaters\_df.set\_index(['title', 'year'], inplace=True)"

]

},

{

"cell\_type": "code",

"execution\_count": 120,

"metadata": {},

"outputs": [],

"source": [

"theaters\_and\_awards = theaters\_df.join(awards\_df, how='inner', on=['title', 'year'])"

]

},

{

"cell\_type": "code",

"execution\_count": 121,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>max\_theaters</th>\n",

" <th>total\_dom\_gross($)</th>\n",

" <th>dollars\_per\_theater</th>\n",

" <th>awards\_won</th>\n",

" <th>awards\_nominated</th>\n",

" <th>win\_rate</th>\n",

" </tr>\n",

" <tr>\n",

" <th>studio</th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>Disney</th>\n",

" <td>22</td>\n",

" <td>22</td>\n",

" <td>22</td>\n",

" <td>22</td>\n",

" <td>22</td>\n",

" <td>22</td>\n",

" </tr>\n",

" <tr>\n",

" <th>DreamWorks Distribution</th>\n",

" <td>4</td>\n",

" <td>4</td>\n",

" <td>4</td>\n",

" <td>4</td>\n",

" <td>4</td>\n",

" <td>4</td>\n",

" </tr>\n",

" <tr>\n",

" <th>New Line Cinema</th>\n",

" <td>2</td>\n",

" <td>2</td>\n",

" <td>2</td>\n",

" <td>2</td>\n",

" <td>2</td>\n",

" <td>2</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Paramount Pictures</th>\n",

" <td>7</td>\n",

" <td>7</td>\n",

" <td>7</td>\n",

" <td>7</td>\n",

" <td>7</td>\n",

" <td>7</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Sony</th>\n",

" <td>6</td>\n",

" <td>6</td>\n",

" <td>6</td>\n",

" <td>6</td>\n",

" <td>6</td>\n",

" <td>6</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Twentieth Century Fox</th>\n",

" <td>4</td>\n",

" <td>4</td>\n",

" <td>4</td>\n",

" <td>4</td>\n",

" <td>4</td>\n",

" <td>4</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Universal Pictures</th>\n",

" <td>6</td>\n",

" <td>6</td>\n",

" <td>6</td>\n",

" <td>6</td>\n",

" <td>6</td>\n",

" <td>6</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Warner Bros.</th>\n",

" <td>15</td>\n",

" <td>15</td>\n",

" <td>15</td>\n",

" <td>15</td>\n",

" <td>15</td>\n",

" <td>15</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" max\_theaters total\_dom\_gross($) \\\n",

"studio \n",

"Disney 22 22 \n",

"DreamWorks Distribution 4 4 \n",

"New Line Cinema 2 2 \n",

"Paramount Pictures 7 7 \n",

"Sony 6 6 \n",

"Twentieth Century Fox 4 4 \n",

"Universal Pictures 6 6 \n",

"Warner Bros. 15 15 \n",

"\n",

" dollars\_per\_theater awards\_won awards\_nominated \\\n",

"studio \n",

"Disney 22 22 22 \n",

"DreamWorks Distribution 4 4 4 \n",

"New Line Cinema 2 2 2 \n",

"Paramount Pictures 7 7 7 \n",

"Sony 6 6 6 \n",

"Twentieth Century Fox 4 4 4 \n",

"Universal Pictures 6 6 6 \n",

"Warner Bros. 15 15 15 \n",

"\n",

" win\_rate \n",

"studio \n",

"Disney 22 \n",

"DreamWorks Distribution 4 \n",

"New Line Cinema 2 \n",

"Paramount Pictures 7 \n",

"Sony 6 \n",

"Twentieth Century Fox 4 \n",

"Universal Pictures 6 \n",

"Warner Bros. 15 "

]

},

"execution\_count": 121,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"theaters\_and\_awards.groupby('studio').count()"

]

},

{

"cell\_type": "code",

"execution\_count": 122,

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>max\_theaters</th>\n",

" <th>total\_dom\_gross($)</th>\n",

" <th>dollars\_per\_theater</th>\n",

" <th>awards\_won</th>\n",

" <th>awards\_nominated</th>\n",

" <th>win\_rate</th>\n",

" </tr>\n",

" <tr>\n",

" <th>studio</th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" <th></th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>Disney</th>\n",

" <td>3818.73</td>\n",

" <td>305217242.45</td>\n",

" <td>78797.61</td>\n",

" <td>1.36</td>\n",

" <td>3.00</td>\n",

" <td>0.60</td>\n",

" </tr>\n",

" <tr>\n",

" <th>DreamWorks Distribution</th>\n",

" <td>3444.25</td>\n",

" <td>153223630.75</td>\n",

" <td>44447.63</td>\n",

" <td>2.00</td>\n",

" <td>4.25</td>\n",

" <td>0.60</td>\n",

" </tr>\n",

" <tr>\n",

" <th>New Line Cinema</th>\n",

" <td>3662.50</td>\n",

" <td>358408603.00</td>\n",

" <td>97814.75</td>\n",

" <td>6.50</td>\n",

" <td>8.50</td>\n",

" <td>0.67</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Paramount Pictures</th>\n",

" <td>3564.86</td>\n",

" <td>140835427.57</td>\n",

" <td>38930.82</td>\n",

" <td>1.00</td>\n",

" <td>3.71</td>\n",

" <td>0.45</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Sony</th>\n",

" <td>3653.67</td>\n",

" <td>237842295.67</td>\n",

" <td>64720.23</td>\n",

" <td>1.17</td>\n",

" <td>3.17</td>\n",

" <td>0.54</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Twentieth Century Fox</th>\n",

" <td>3501.75</td>\n",

" <td>136874930.25</td>\n",

" <td>38404.79</td>\n",

" <td>2.25</td>\n",

" <td>6.00</td>\n",

" <td>0.43</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Universal Pictures</th>\n",

" <td>3338.83</td>\n",

" <td>149344665.00</td>\n",

" <td>44970.82</td>\n",

" <td>1.33</td>\n",

" <td>3.33</td>\n",

" <td>0.51</td>\n",

" </tr>\n",

" <tr>\n",

" <th>Warner Bros.</th>\n",

" <td>3831.60</td>\n",

" <td>234055876.80</td>\n",

" <td>60023.04</td>\n",

" <td>2.67</td>\n",

" <td>5.87</td>\n",

" <td>0.56</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" max\_theaters total\_dom\_gross($) \\\n",

"studio \n",

"Disney 3818.73 305217242.45 \n",

"DreamWorks Distribution 3444.25 153223630.75 \n",

"New Line Cinema 3662.50 358408603.00 \n",

"Paramount Pictures 3564.86 140835427.57 \n",

"Sony 3653.67 237842295.67 \n",

"Twentieth Century Fox 3501.75 136874930.25 \n",

"Universal Pictures 3338.83 149344665.00 \n",

"Warner Bros. 3831.60 234055876.80 \n",

"\n",

" dollars\_per\_theater awards\_won awards\_nominated \\\n",

"studio \n",

"Disney 78797.61 1.36 3.00 \n",

"DreamWorks Distribution 44447.63 2.00 4.25 \n",

"New Line Cinema 97814.75 6.50 8.50 \n",

"Paramount Pictures 38930.82 1.00 3.71 \n",

"Sony 64720.23 1.17 3.17 \n",

"Twentieth Century Fox 38404.79 2.25 6.00 \n",

"Universal Pictures 44970.82 1.33 3.33 \n",

"Warner Bros. 60023.04 2.67 5.87 \n",

"\n",

" win\_rate \n",

"studio \n",

"Disney 0.60 \n",

"DreamWorks Distribution 0.60 \n",

"New Line Cinema 0.67 \n",

"Paramount Pictures 0.45 \n",

"Sony 0.54 \n",

"Twentieth Century Fox 0.43 \n",

"Universal Pictures 0.51 \n",

"Warner Bros. 0.56 "

]

},

"execution\_count": 122,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"theaters\_and\_awards.groupby('studio').mean()"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Unfortunately, the joining of the dataframes only left us with 66 common movies. We would prefer to have more data to be more confident in establishing trends. We will consider the average number of theaters and average win rate to make a determination. Disney is associated with 22 movies in our joined dataframe while Warner Bros. is associated with 15. Warner. Bros does have a higher average for the number of theaters, however Disney has a noticeable $18,000 advantage in average domestic gross per theater. Disney also has the higher win rate for Oscars at nearly 60%.\n",

"\n",

"\*\*Question 8 Conclusion\*\*: Our Company should research Disney's best practices and try to build off the success of this well established studio."

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"# Conclusion\n",

"While there are many other factors that we could consider in a future analysis we feel that the following 8 conclusions will result in a successful business venture as our Comapany enters the movie industry.\n",

"\n",

"1. I recommend that we should budget approximately \\$82,250,000 to make a movie. This should correlate with a profit margin above 80\\%.\n",

"2. I recommend that we should focus their efforts on the top 6 most profitable movie genres: Adventure, Action, Comedy, Drama, Sci-Fi and Animation. A further recommendation to focus on Sci-Fi and Animation due to less competition and a higher opportunity to profit.\n",

"3. I recommend that we release the bulk of their movies, especially Animation, during the summer months. Adventure, Drama and Comedy movies would see similar success if released in November, but the recommendation remains to focus on summer.\n",

"4. Question 4 Conclusion: I recommend that we focus their cast and crew search to individuals who consistently score at least 1.0 on the VAR score. We can, with a high level of confidence, conclude that these individuals will elevate the overall production.\n",

"5. We should spend at least \\$35,465,000 in order to make an Oscar-winning movie.\n",

"6. I recommend that we take into consideration the rating of the movie based on the genre and target audience. If making animation movies, it is wise to stick to a G or PG rating, otherwise PG-13 is the sweetspot. In terms of runtime, there is little correlation in terms of overall profitability.\n",

"7. We should aim for a profit margin of 66% and a net profit of slightly over 50 million per movie to compete with the top existing studios.\n",

"8. We should research Disney's best practices and try to build off the success of this well established studio."

]

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3 (ipykernel)",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.9.2"

},

"toc": {

"base\_numbering": 1,

"nav\_menu": {},

"number\_sections": false,

"sideBar": true,

"skip\_h1\_title": false,

"title\_cell": "Table of Contents",

"title\_sidebar": "Contents",

"toc\_cell": false,

"toc\_position": {},

"toc\_section\_display": true,

"toc\_window\_display": false

},

"varInspector": {

"cols": {

"lenName": 16,

"lenType": 16,

"lenVar": 40

},

"kernels\_config": {

"python": {

"delete\_cmd\_postfix": "",

"delete\_cmd\_prefix": "del ",

"library": "var\_list.py",

"varRefreshCmd": "print(var\_dic\_list())"

},

"r": {

"delete\_cmd\_postfix": ") ",

"delete\_cmd\_prefix": "rm(",

"library": "var\_list.r",

"varRefreshCmd": "cat(var\_dic\_list()) "

}

},

"types\_to\_exclude": [

"module",

"function",

"builtin\_function\_or\_method",

"instance",

"\_Feature"

],

"window\_display": false

}

},

"nbformat": 4,

"nbformat\_minor": 2

}