Ai Academy Capstone Group 4

Presented by Brian Gorbea, Kelvin Cupay, Robert Cofer, Rob Masters, Ryan Lazar

Created & Consolidated by Kelvin using notebooks of group members

Notebook Structure

1. Overview
2. Business Understanding

- Include stakeholder and key business questions

3. Data Understanding and Analysis

- Source of data
- Description of data
- Three visualizations (the same visualizations presented in the slides and notebook)

4. Statisical Communication & Data Analysis

- Results of statistical inference
- Interpretation of these results in the context of the problem
- Hypothesis testing

5. Conclusion

- Summary of conclusions including three relevant findings
- Future Works

Overview

Using Explanatory data analysis(EDA) and statistical methods we are going to advice Computing Vision how the best way to jump start their new movie studio, called 'Computing Vision Entertainment'. We used sqlite3 in pandas to query the datasets given and hypothesis testing to evaluate our findings. Essentially, we are generating insights for their business and stake holders by making a two part suggestion, one is the early stage of the movie studio and scalability after several years.

Business Understanding:

Goals: The business question we are going to answer Assumption : The parent company 'Computing Vision' does not have much backgrounk in creating movies.

1. Best investment in order to quick start the revenue of new movie studio 'Computing Vidion Entertainment'
2. Best investment in order to scale the business and future ventures in the business industry

This notebook will drive deeper into how we utilize the data sets to create meaningful visualizations and aggregrations in order to achieve our goals listed above.

We will also create hypothesis test to evaluate our findings and reccomendations which are based in scietific & mathematical reasonings. The reccomendation for the *head of Computing Vision's new movie studio* and help him decide which films to create the best film in terms of profits and appreciation.

Real World Application:

Today less people are going to the theatres and watching movies. Maybe Hollywood has gone stale and have stagnanted with their ways of creating movies, through our insights of what is most profitable for a new and small scale movie studio someone can use this information to create and start their new movie business. `Note:` that our client is a tech company, this may hinder start-ups.

Stakeholders:

- `Investors` : Can identify by profits if rate on investment(ROI) is significant for them to invest in Computing Vision Entertainment
- `Computing Vision(Parent Company)` : Is it worth it to make
- `Critics` : Is this new movie studio just for profits or they actually want to create movies that are well liked
- `Fans` : Do we like the movies from this mobie studio
- `Potential future fans` : Repuation of this new in coming movie studio

Data Undertanding:

Description of data:

The data set comes from several sources where some are compressed into CSV(comma-separated values) or TSV(tab-separated values). The data sets listed below are from these respectable sources: Box Office Mojo (https://www.boxofficemojo.com/), IMDB (https://www.imdb.com/), Rotten Tomatoes (https://www.rottentomatoes.com/), The MovieDB (https://www.themoviedb.org/), The Numbers (https://www.the-numbers.com/).

Here is a list of the data sets use:

- `bom.movie_gross.csv.gz` :
- `im.db.zip` :
- `rt.movie_info.tsv.gz` :
- `rt.reviews.tsv.gz` :
- `tmdb.movies.csv.gz` :
- `tn.movie_budgets.csv.gz:`

Importing Libraries used

In [1]:
```
# imports
import pandas as pd
import numpy as np
import scipy as sp
import pandas as pd
import scipy.stats as stats
import seaborn as sns
import matplotlib.pyplot as plt
import math
import sqlite3
import zipfile
%matplotlib inline
```

In [2]:
```
#add data / making dataframes
movie_gross_df = pd.read_csv('../../data/bom.movie_gross.csv.gz')
movie_info_df = pd.read_csv('../../data/rt.movie_info.tsv.gz', sep='\t')
reviews_df = pd.read_csv('../../data/rt.reviews.tsv.gz', delimiter='\t', en
movies_df = pd.read_csv('../../data/tmdb.movies.csv.gz', encoding='latin1')
movie_budgets_df = pd.read_csv('../../data/tn.movie_budgets.csv.gz', encodi
```

Data Organizing, Data Merging, Data Formatting, Data Calculation

Change the column "title in the bom.movie dataframe. This will then allow us to merge both dataframes on that column.

After the column name was changed we can now merge and begin our data exploration.

Before we can work with the numbers in the columns we have to remove the $ and any spaces that may exist.

Now we create a profit column in the dataframe and we calculate that by subtracting the "production_budget" column from the "worldwide_gross" column. It is a large number so we then divide by 1,000,000 to change it out of scientific notation.

In [3]:
```
# Rename the title column to movie for merge with the movie_budget datafram
movie_gross_df = movie_gross_df.rename(columns={'title': 'movie'})
```

In [4]:
```
#connect moviegross and moviebudgets on movie
gross_budgets = pd.merge(movie_gross_df, movie_budgets_df, on= 'movie')
```

In [5]:
```
# Have to reformat the worldwide_gross and production_budget columns to rem
gross_budgets['worldwide_gross'] = gross_budgets['worldwide_gross'].replace
gross_budgets['production_budget'] = gross_budgets['production_budget'].rep
```

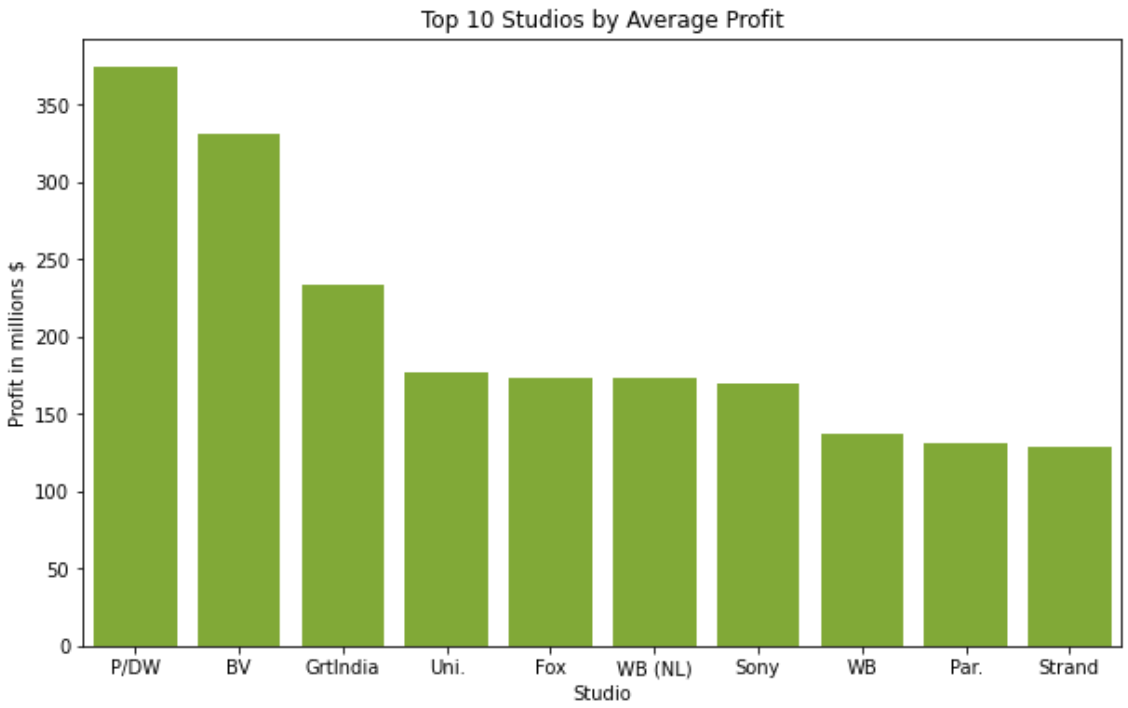
In [6]:
```
# Create a profit column by subtracting production_budget from worldwide_gr
gross_budgets['profit'] = (gross_budgets['worldwide_gross'] - gross_budgets
```

First Visualization

This graph shows the top 10 studios that have the highest profit on average. It was created by combing the two dataframes movie_gross_df and movie_budgets_df, calculating profit, and then grouping by the studio. The values are the average profit (mean).

In [7]:
```
# Plot top 10 - Cleaned up
plt.figure(figsize=(10, 6))
ax = sns.barplot(x='studio', y='profit', data=gross_budgets.groupby('studio
plt.ylabel('Profit in millions $')
plt.xlabel('Studio')
plt.title('Top 10 Studios by Average Profit')
```

Out[7]:
```
Text(0.5, 1.0, 'Top 10 Studios by Average Profit')
```



Ryan Lazar's Code

Brian Gorbea's Code

Importing and starting the sqlite

In [8]:
```
# Path to the zip file
zip_file_path = '../../data/im.db.zip'
# Folder to extract the zip file contents
extract_folder = '../../data'
```

The process here is pulling the databse file, unzipping the file and extracting the file to a local location. The file is then opened and read using SQL commands within python. Python pulls the data and stores it in the a variable called a dataframe for later use.

In [9]:
```
# Open the zip file
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    # Extract the contents of the zip file to the extract_folder
    zip_ref.extractall(extract_folder)
```

In [10]:
```
db_file_path = f'{extract_folder}/im.db'
conn = sqlite3.connect(db_file_path)
cursor = conn.cursor()
```

In [11]:
```
q = """
SELECT primary_title, genres
FROM movie_basics
;"""
movie_genres = pd.read_sql_query(q, conn)
```

Start of data pulling and organization

Below the data is being pulled from each file and being stored to a variable for use.

The genre data in the database stored more than one genre per movie. The solution was to seperate the genres and pull the first genre that appears and mark that as the primary genre.

These are using sql so df was restablished

In [12]:
```
movie_genres['primary_genre'] = movie_genres['genres'].str.split(',').str.g
```

In [13]:
```
movie_genres = movie_genres.rename(columns={'primary_title': 'movie'})
```

In [14]:
```
movie_budgets_df = pd.read_csv('../../data/tn.movie_budgets.csv.gz', encodi
```

In [15]:
```
movie_gross_df = pd.read_csv('../../data/bom.movie_gross.csv.gz')
```

In [16]:
```
movie_gross_df = movie_gross_df.rename(columns={'title': 'movie'})
```

In [17]:
```
Bugets_Gross_df = pd.merge(movie_gross_df, movie_budgets_df, on=['movie'])
```

Cleaning the gross and budget data

The steps below looks at the columns for world wide gross and production budgets and turn them into readable values that python can use to do calculations. This way we can easily create the profit table to use for the vizualizations.

In [18]:
```
# Clean the worldwide_gross column
Bugets_Gross_df['worldwide_gross'] = Bugets_Gross_df['worldwide_gross'].str
```

In [19]:
```
# Convert the column to float
Bugets_Gross_df['worldwide_gross'] = Bugets_Gross_df['worldwide_gross'].ast
```

In [20]:
```
# Clean the production_budget column
Bugets_Gross_df['production_budget'] = Bugets_Gross_df['production_budget']

# Convert the column to float
Bugets_Gross_df['production_budget'] = Bugets_Gross_df['production_budget']
```

In [21]:
```
# Calculate the profit column
Bugets_Gross_df['profit'] = Bugets_Gross_df['worldwide_gross'] - Bugets_Gro
```

Sorting data

The data is being sorted by most profitable to least profitable. One it has been sorted the data grabs the top 10 results from the list. The top ten list is then used for vizualiation later for top ten profitable movies.

In [22]:
```
profit_sorted= Bugets_Gross_df.sort_values(by='profit', ascending=False)
topTen = profit_sorted.head(10)
```

In [23]:
```

topTen_rev = topTen.iloc[::-1].reset_index(drop=True)
topTen_rev['profit'] = topTen_rev['profit']/1000000
```

This is the end of the major data collection

The base data has been sorted.

In [24]:
```
# takes in values and reformats values (handles negative values)
def format_values(value):
    if value >= 1000000000: # Billions
        return f"{value / 1000000000:.1f}B"
    elif value >= 1000000: # Millions
        return f"{value / 1000000:.1f}M"
    elif value < 0 and abs(value) >= 1000000: # Negative Millions
        return f"{value / 1000000:.1f}M"
    elif value >= 1000: # Thoudsands
            return f"{value / 1000:.1f}K"
    else:
        return str(value)
```

In [25]:
```
# takes in values and reformats values
# these are used in the bargraphs
# handles x values
def format_values_x(value):
    if value >= 1000000000: # Billions
        return f"{value / 1000000000:.1f}"
    elif value >= 1000000: # Millions
        return f"{value / 1000000:.1f}"
    elif value <0 and abs(value) >= 1000000: # Negative Millions
        return f"{value / 1000000:.1f}"
    else:
        return str(value)
```

In [26]:
```
# sorting profit from profitable to least profitable(descending order)
profit_sorted= gross_budgets.sort_values(by='profit', ascending=False)
```

Creating Data

Data is being merged and calculated to represent the average based on the primary genre of a movie.

In [27]:
```
# merging movies genres and profit sorted on movie
profgenre = pd.merge(movie_genres, profit_sorted, on=['movie'])
```

In [28]:
```
# grouping by primary genre and getting mean then sorting by profit
profgenre_grouped = profgenre.groupby('primary_genre').mean()['profit'].sor
```

Second Visualization - Top Ten Profitable Genres with a budget Under 10M

The vizualization is used to get the genre with a budget under 10m. This data can represent which genres don't do well and what to potentially avoid.

In [29]:
```
# merge budget gross and movie genres on movie
budget_UTen = pd.merge(Bugets_Gross_df,movie_genres, on = 'movie')
```

The data is merged on movie names then sorted to only include production budgets under 10m. Once the data has been sorted the mean value for the primary genre is calculated and is sorted based on the total profit. What this produces is a dataframe with two columns. The first column is the name of the genre and the second column is the averaged profit that genre made in the past 50 years while also maintaing a budget under 10m.

In [30]:
```
# selecting production budget under 10 mill
budget_UTen = budget_UTen[budget_UTen['production_budget']< 10000000]
```

In [31]:
```
# grouping by primary genre then getting mean  then sorting by profit
budget_UTen_mean =budget_UTen.groupby('primary_genre').mean()['profit'].sor
```

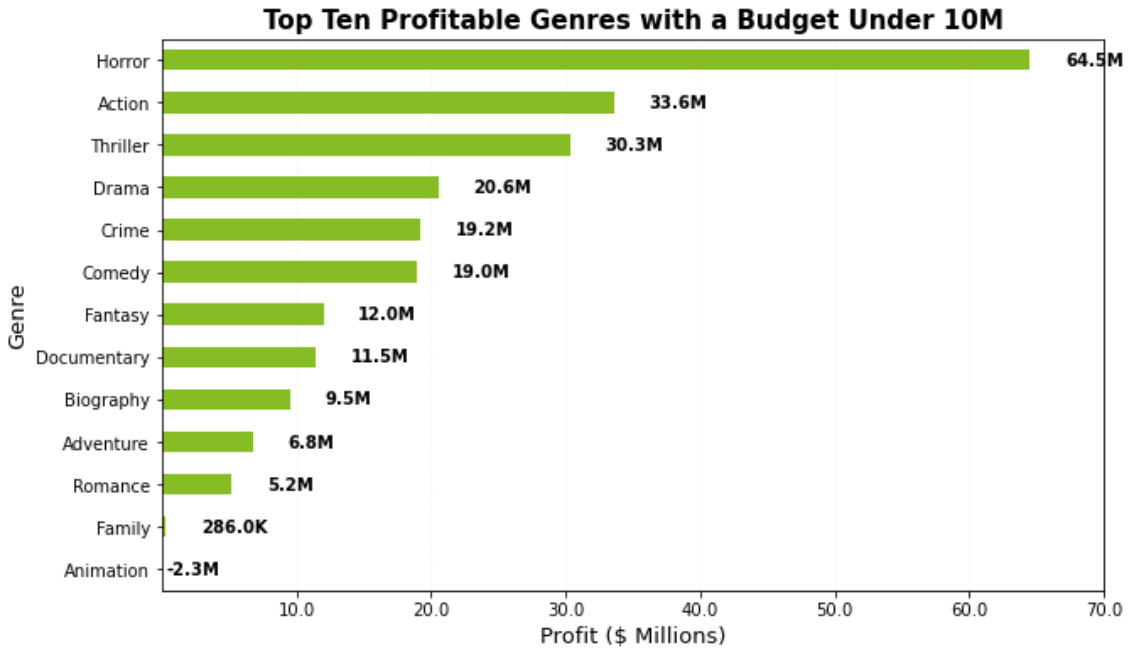
In [32]:

```
#Plotting
fig, ax = plt.subplots(figsize=(10,6))
budget_UTen_mean.plot(kind='barh',color='#86BC24', ax=ax)

#Title and labels
ax.set_title('Top Ten Profitable Genres with a Budget Under 10M', fontweigh
ax.set_xticks(ax.get_xticks())
ax.set_xticklabels([format_values_x(x) for x in ax.get_xticks()])
ax.set_xlabel('Profit ($ Millions)', fontsize=13)
ax.set_ylabel('Genre', fontsize=13)
ax.grid(axis='x', linestyle='--', alpha = 0.07)
ax.set_xlim(1, ax.get_xlim()[1])

#for index, value in enumerate(budget_UTen_mean):
#    formatted_value = "${:.1f}".format(value)
#    ax.text(value , index, ' ' + str(formatted_value), va='center', fontwe
for index, value in enumerate(budget_UTen_mean):
    ax.text(value + (0.04 * max(budget_UTen_mean)), index,format_values(val


plt.tight_layout
plt.show()
```



Third Visualization - Top Average Profits by Genre

The vizualization is used to get the average profit based on genre. This data can represent which genres don't do well and what to potentially avoid. Such as making a Western movie may not produce results.

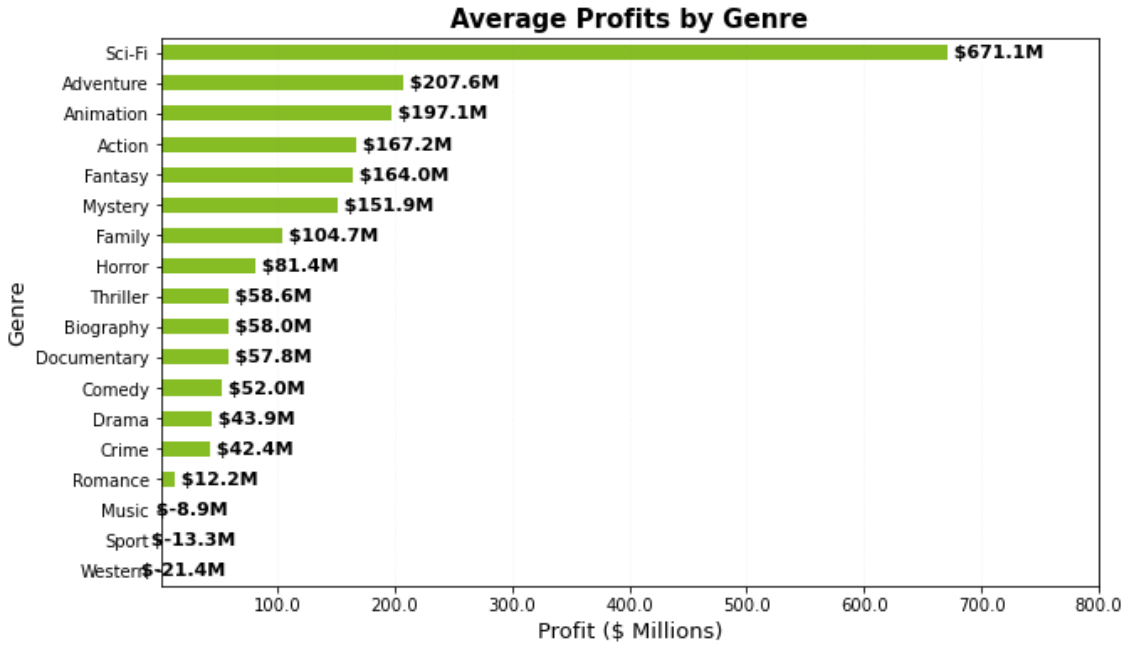
In [33]:

```
#Plotting
fig, ax = plt.subplots(figsize=(10,6))
profgenre_grouped.plot(kind='barh', color="#86BC24", ax=ax)

#Title and labels
ax.set_title('Average Profits by Genre', fontweight='bold', fontsize=15)
ax.set_xticks(ax.get_xticks())
ax.set_xticklabels([format_values_x(x) for x in ax.get_xticks()])
ax.set_xlabel('Profit ($ Millions)', fontsize=13)
ax.set_ylabel('Genre', fontsize=13)
ax.grid(axis='x', linestyle='--', alpha = 0.07)
ax.set_xlim(1, ax.get_xlim()[1])


for index, value in enumerate(profgenre_grouped):
    formatted_value = "${:.1f}M".format(value)
    ax.text(value, index, ' ' + str(formatted_value), va='center', fontweig

plt.tight_layout
plt.show()
```



Brian Gorbea Code

Statisical Communication & Data Analysis

Rob Master's Code

In [34]:
```
#rename dfs to connect on the unique ID movie
movies_df = movies_df.rename(columns={'original_title': 'movie'})
movie_gross_df = movie_gross_df.rename(columns={'title': 'movie'})
```

In [35]:
```
#connect moviegross and moviebudgets on movie
gross_budgets = pd.merge(movie_gross_df, movie_budgets_df, on=['movie'])
```

Below we clean out the new dataframe gross budgets to modify the worldwide_gross and production budget values to ensure we can calculate profit. This starts by removing '$' and ',' from the cells and then converting the data into float.

In [36]:
```
#cleaned for profit and hypothesis testing

# Clean the worldwide_gross column
gross_budgets['worldwide_gross'] = gross_budgets['worldwide_gross'].str.rep

# Convert the column to float
gross_budgets['worldwide_gross'] = gross_budgets['worldwide_gross'].astype(

# Clean the production_budget column
gross_budgets['production_budget'] = gross_budgets['production_budget'].str

# Convert the column to float
gross_budgets['production_budget'] = gross_budgets['production_budget'].ast
```

Here we do the same thing, except this is just for the frequency analysis. Since we dont need studio information, it is better that we use just the movie_budgets_df dataframe as we get better quality data in regards to profits as there isn't data lost from the merge.

In [37]:
```
#Movie budget DF used as there is more data for the frequency analysis for

# Clean the worldwide_gross column
movie_budgets_df['worldwide_gross'] = movie_budgets_df['worldwide_gross'].s

# Convert the column to float
movie_budgets_df['worldwide_gross'] = movie_budgets_df['worldwide_gross'].a

# Clean the production_budget column
movie_budgets_df['production_budget'] = movie_budgets_df['production_budget

# Convert the column to float
movie_budgets_df['production_budget'] = movie_budgets_df['production_budget
```

Here we calculate profits and change the vaules of profit and production budget to be represented in the millions. This is just for the movie budget dataframe.

In [38]:
```
#create profit
movie_budgets_df['profit'] = movie_budgets_df['worldwide_gross'] - movie_bu
#represented in millions
movie_budgets_df['profit'] = movie_budgets_df['profit'] / 1000000
```

In [39]:
```
#create profit
gross_budgets['profit'] = gross_budgets['worldwide_gross'] - gross_budgets[
#represented in millions
gross_budgets['profit'] = gross_budgets['profit'] / 1000000
```

Analysis on profit

Here we do a general analysis about profits. We use this dataframe because it has better data quality.

In [40]:
```
#stat info on profit
profit_stats = movie_budgets_df['profit'].describe()
print(profit_stats)
```

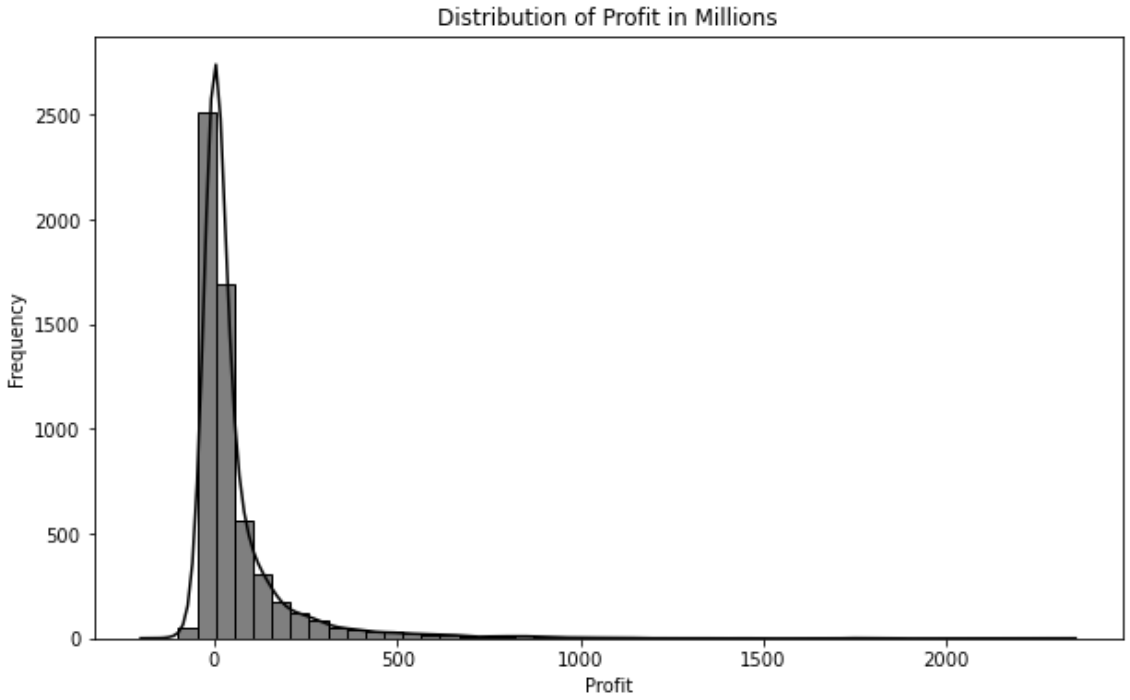
```
count    5782.000000
mean       59.899704
std       146.088881
min      -200.237650
25%        -2.189071
50%         8.550286
75%        60.968502
max      2351.345279
Name: profit, dtype: float64
```

Creating histogram for profit

Below we create a histogram for profit. This provides us with a general anaysis of where profit lines up for all films, generally showing us the types of returns we can expect.

In [41]:

```
# Draw a histogram for profit
plt.figure(figsize=(10, 6))
sns.histplot(data=movie_budgets_df, x='profit', kde=True, bins = 50,
             color='black')
plt.xlabel('Profit')
plt.ylabel('Frequency')
plt.title('Distribution of Profit in Millions')
plt.show()
```



Null Hypothesis:

Typically there is no relationship between A and B. In our case, profit has no relationship to any of the top 3 occuring studios.

Alternative Hypothesis:

The alternative hypothesis is traditionally thought of when creating a hypothesis for an experiment. In our case, profit has a relationship to all or some of the top 3 occuring studios.

General anaylsis for developing hypothesis testing

Here we check for all unique studios in order to get an idea of all the potential studios to emulate.

In [42]:
```
#check all the unique studios
studio_list = gross_budgets['studio'].unique()
studio_list
```

Out[42]:
```
array(['BV', 'WB', 'P/DW', 'Sum.', 'Par.', 'Uni.', 'Fox', 'Sony', 'FoxS',
       'SGem', 'WB (NL)', 'LGF', 'MBox', 'W/Dim.', 'Focus', 'MGM',
       'Over.', 'Mira.', 'NM', 'CBS', 'SPC', 'ParV', 'Gold.', 'RAtt.',
       'Magn.', 'IFC', 'Free', '3D', 'Wein.', 'Rela.', 'Anch.', 'App.',
       'Drft.', 'IW', 'Relbig.', 'Viv.', 'Eros', 'Scre.', 'UTV', 'Kino',
       'ATO', 'First', 'GK', 'NFC', 'Strand', 'Mont.', 'IVP', 'FD',
       'TriS', 'ORF', 'Jan.', 'Osci.', 'OMNI/FSR', nan, 'SMod', 'WHE',
       'P4', 'ALP', 'LG/S', 'RTWC', 'MNE', 'LD', 'Yash', 'IM', 'A24',
       'PH', 'EOne', 'ELS', 'CE', 'Saban', 'DR', 'Trib.', 'KE', 'VE',
       'EC', 'BG', 'PFR', 'BST', 'BH Tilt', 'BSC', 'FCW', 'Cohen', 'LGP',
       'TFA', 'Alc', 'STX', 'Orch.', 'PNT', 'CJ', 'Cleopatra', 'BBC',
       'GrtIndia', 'Neon', 'Affirm', 'ENTMP', 'Studio 8', 'Annapurna',
       'Global Road', 'Amazon', 'RLJ'], dtype=object)
```

Here we check to see the total count of each studio. We are focusing on high studio frequency in the dataset to get an accurate pvalue test, as studios like Trib or FCW will not provide us with accurate results

In [43]:
```
#check to see the total count of each studio
#low studio frequency will not be accurately represented with the data so w
studio_counts = gross_budgets['studio'].value_counts()
print(studio_counts)
```

```
Uni.     117
Fox      110
WB       102
Par.      74
Sony      74
        ...
3D         1
BBC        1
Viv.       1
ELS        1
GK         1
Name: studio, Length: 99, dtype: int64
```

Hypothesis testing

In this code we run a loop to check 3 different top occuring studios independently to see if they are significant to profit. We create a place to store the studios, and then create the loop which counts studios, sorts them by an occurrence count in decending order and then grabs the top 3 to do a p-value test independently of each other for studio and profit. One other condition we check for is if there is a null value for studio and drop it from the results.

In [44]:

```
#in this code we run a loop to check 3 different top occuring studios
# to see if they are significant to profit

#place to store studios
studio_counts = {}
studio_list = gross_budgets['studio'].unique()

for studio in studio_list:
    #count studios
    count = len(gross_budgets[gross_budgets['studio'] == studio])
    studio_counts[studio] = count

# Sort the dictionary by values (occurrence count) in descending order
sorted_studios = sorted(studio_counts.items(), key=lambda x: x[1], reverse=

print("Top 3 studio occurances in the dataset significance with profit:")
#grab the top 10
for studio, count in sorted_studios[:3]:
    #create var for pvalue with respect to studio and profit
    studio_profit = gross_budgets[gross_budgets['studio'] == studio]['profi
    #pvalue test
    t_statistic, p_value = stats.ttest_1samp(studio_profit, popmean=0)
    # Check if p-value is not null
    if not np.isnan(p_value):
        #then print the results
        print(f"{studio}: {p_value}")
```

```
Top 3 studio occurances in the dataset significance with profit:
Uni.: 3.688020261558896e-10
Fox: 9.610055205557015e-17
WB: 4.328957530102738e-10
```

Rejecting the null hypothesis:

Based on the results of each of the three studios being significant at the 95% threshold, we reject the null and accept the alternative that any of the top 3 occuring studios will help with profit

Confidence Intervals

Here in the code below we check the confidence interval for profit for films with a production budget under 10 million dollars.

In [45]:
```
# Create a new dataframe with production budgets under 10 million
production_budget_under_10m = gross_budgets[gross_budgets['production_budge

confidence_level = 0.95
sample_size = len(production_budget_under_10m['profit'])

mean_profit = np.mean(production_budget_under_10m['profit'])
std_dev = np.std(production_budget_under_10m['profit'])
std_error = stats.sem(production_budget_under_10m['profit'])

margin_of_error = std_error * stats.t.ppf((1 + confidence_level) / 2, sampl

lower_bound_low10 = round(mean_profit - margin_of_error, 2)
upper_bound_low10 = round(mean_profit + margin_of_error, 2)

print(f"The {confidence_level * 100}% confidence interval for movie budgets
```

```
The 95.0% confidence interval for movie budgets under 10 million represent
ed in millions is: (15.59, 26.02)
```

We do the same thing as above but for films with a production budget over 10 million dollars.

In [46]:
```
# Create a new dataframe with production budgets greater than or equal to 1
production_budget_over_10m = gross_budgets[gross_budgets['production_budget
confidence_level = 0.95
sample_size = len(production_budget_over_10m['profit'])

mean_profit = np.mean(production_budget_over_10m['profit'])
std_dev = np.std(production_budget_over_10m['profit'])
std_error = stats.sem(production_budget_over_10m['profit'])

margin_of_error = std_error * stats.t.ppf((1 + confidence_level) / 2, sampl

lower_bound_up10 = round(mean_profit - margin_of_error, 2)
upper_bound_up10 = round(mean_profit + margin_of_error, 2)

print(f"The {confidence_level * 100}% confidence interval for movie budgets
```

```
The 95.0% confidence interval for movie budgets under 10 million represent
ed in millions is: (115.68, 142.52)
```

Rob Master's Code

Conclusion

Business Recommendations

When we started to work and explored data we thought it was Computing Vision's best interest to have a `2 phase` recommendation. The first one being the short term phase(0-5 years) which consist of limiting capital expenditure to under 10 million dollars. As we gain familiarity of the movie industry (5-10 years in the future) and grow our entertainment portfolio, we can expand into higher grossing films such as creating movies over 10 million dollar production budget.

We chose 10 million as a threshold number since we strongly believe that this is feasible capital expenditure for a tech company like Computing Vision. We also saw that 10 million is 25th percentile above and below 50th in production budget which makes it safe and retains market viability.

This provides us the general statistical analysis in order to justify a 10 million dollar production budget limit for a new movie studio. This would put us above the bottom quarter but keep our movies viable with the broader film market.

In [47]:
```
budget_stats = movie_budgets_df['production_budget'].describe()
print(budget_stats)
```

```
count    5.782000e+03
mean     3.158776e+07
std      4.181208e+07
min      1.100000e+03
25%      5.000000e+06
50%      1.700000e+07
75%      4.000000e+07
max      4.250000e+08
Name: production_budget, dtype: float64
```

1. Short Term Reccommendation

Make `horror movies` with a `10 million budget or less` in the short-term. The `genre itself has the highest profits on average` for a budget under 10 million and also consists of 9 out of the top 10 most profitable low budget films.

2. Growth Phase

Branch into `Sci-Fi` , `Adventure` , and `Animation Films` once you are established. While these are the top profiting genres we do `caution` about high production budgets.

3. Emulating A Studio

Since we have found that they can influence profits, Emulating the right studio is a key to jumpstarting a movie studio based on our findings. The top 3 we recommend are `Pixar` , `BV`

Future Works

The scope of the task allowed only Exploratory Data Analysis(EDA) and Hypothesis Testing. We hope in the future we can use Regression and Machine Learning Algorithms to improve and validate our findings. Here are some of the example of what we would like to do in the future.

1. Look into other variables that have effects on profit (Director, Revenue, Ratings)
2. Regression testing to determine data relationships
3. Predictive modeling with machine learning
4. Perform thorough risk analysis for upcoming projects
5. Risk evaluation on genres

End of Note book: Thank you so much for reading to this point!

In [ ]: