Final Project Submission

Please fill out:

- **#### Student name: Ian Macharia**
- **#### Student pace: full time**
- **#### Scheduled project review date/time:**
- **#### Instructor name:**
- **#### Blog post URL: https://github.com/Imacharia/Microsoft-Movie-Studio-market-analysis**

Movie studio analysis

Importing relevant packages

The packages we use are the built upon base Python language. They include: `Numpy` Package for mathematical analysis if we will need `Pandas` package - which will be used for cleaning and subsetting the data into dataframe `SQlite3` package for extracting data from databases used. `Matplotlib` package for some basic visualization `Seaborn package` for more detailed visualizations and clearer visualizations. It is common practice to import the packages using their aliases rather than having to call their full names.

note: Within the cells are additional information written using `#` and `""" """` while the major points to note will be in Markdown form as in this case.

In [32]:

```
# importing relevant Packages.

#importing Numpy
import numpy as np

#importing Pandas
import pandas as pd

#importing SQLite3
import sqlite3 as sqlite3

#importing Matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

#importing Seaborn
import seaborn as sns
```

Business Undertanding

The business problem we are presented with is that Microsoft wants to enter the movie industry and create a new movie studio. However, they have limited knowledge about creating movies, and they need to understand the market demand and what types of films are currently successful at the box office.The stakeholders in this project include Microsoft executives, investors, and potential moviegoers who are interested in watching new and exciting movies.

To address this problem, the project must focus on identifying the types of movies that are currently doing well in the market and provide actionable insights to the head of Microsoft's new movie studio. The problem is to find the right strategy for Microsoft's new movie studio by understanding the audience's preferences and current market trends. By identifying the types of films that are currently successful, Microsoft can create a plan that aligns with the market's demand and attract potential customers.

Having a clear undertsnding of this will help Microsoft minimize risks, reduce costs, and increase the chances of success in a highly competitive market.Since the project will create value by assisting Microsoft shareholders make informed decisions by providing actionable insights about the type of films that are currently successful.

Overall, the project's goal is to assist Microsoft in creating a successful movie studio by identifying the types of films that are currently successful and translating these findings intoactionable insights that can guide decision-making.

From the currenlty provided data set of movies, we can explore what type of films are currently doing the best at the box office. some of the columns we might consider are `genres` , `averagerating` , `domestic_gross` , and `worldwide_gross` just to name a few.

First, we can group the movies by `genres` and calculate the `average rating` , `domestic gross` , and `worldwide gross` for each `genre` . This will give us an idea of which genres are more popular among audiences and are generating more revenue. We can use this information to identify the top genres that Microsoft's new movie studio should focus on.

Second, we can analyze the relationship between the movie rating and revenue. This analysis will give us insights into the impact of ratings on box office performance. It will help us understand whether higher-rated movies tend to perform better or not.

Finally, we can also look at the release date of the movies to see if there are any trends or patterns. This analysis can help us identify the best time to release a movie for maximum revenue.

Overall, by analyzing the provided dataset, we can provide actionable insights to Microsoft's new movie studio regarding what types of films to create.

Although there are more than the above mentioned analyzed relationships within this Notebook

Key Business Questions

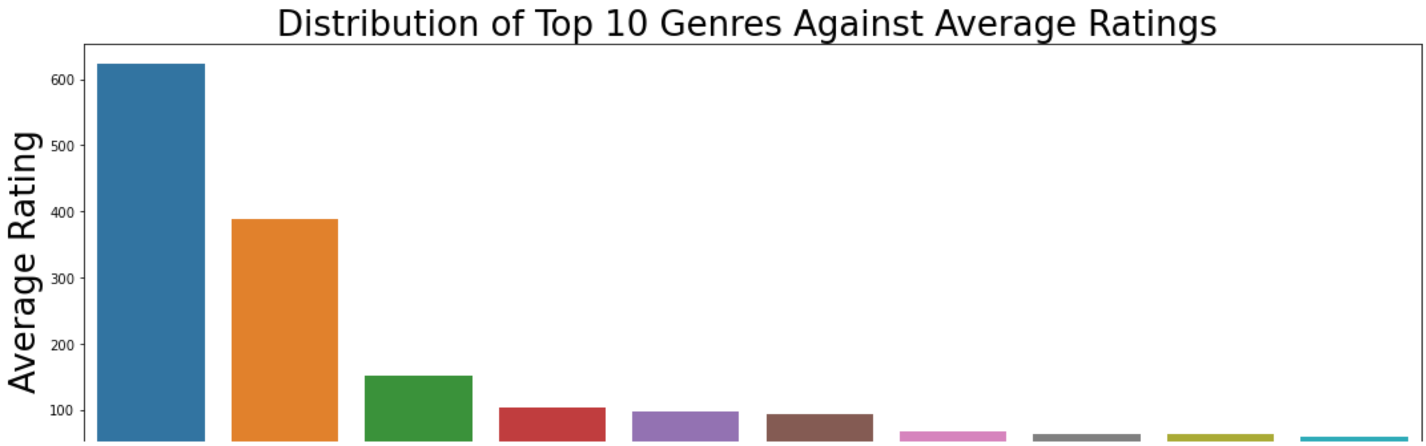
Here are some key business questions that could be addressed through analysis of the provided dataset:

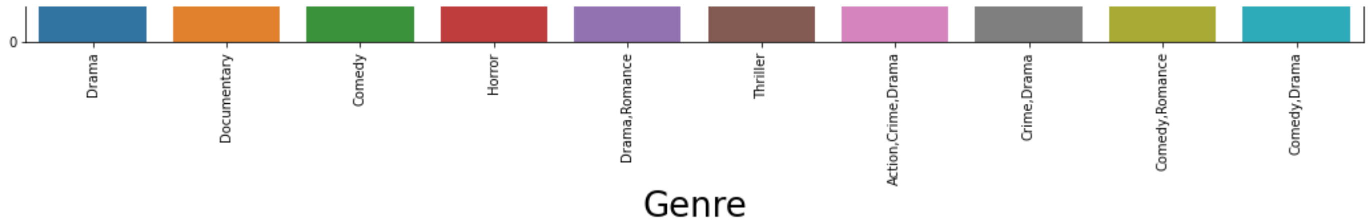
1. What are the most popular movie genres in terms of audience ratings and revenue?

To address this question, we can create a `bar chart` showing the `average rating` , `domestic gross` , `profit` and `worldwide gross` for each movie `genre` .

We can draw a bar graph showing the distribution of average ratings against genre Using the combined data,we observe a very unclear plot is presented. This can easily be corrected by creating a subset from containing only genre and rating this still does not get us the desired results, we therefore sort the data in descending order, and only plot the highest rated movies

 # Create a box plot of average ratings by genre y_to_plot = genre_data['averagerating'].head(10) x_to_plot = genre_data['genres'].head(10) plt.figure(figsize=(18,6)) sns.barplot(x=x_to_plot, y=y_to_plot, data=genre_data) plt.xticks(rotation=90) plt.xlabel('Genre',fontsize=26) plt.ylabel('Average Rating',fontsize=26) plt.title('Distribution of Top 10 Genres Against Average Ratings ',fontsize=26) plt.show()





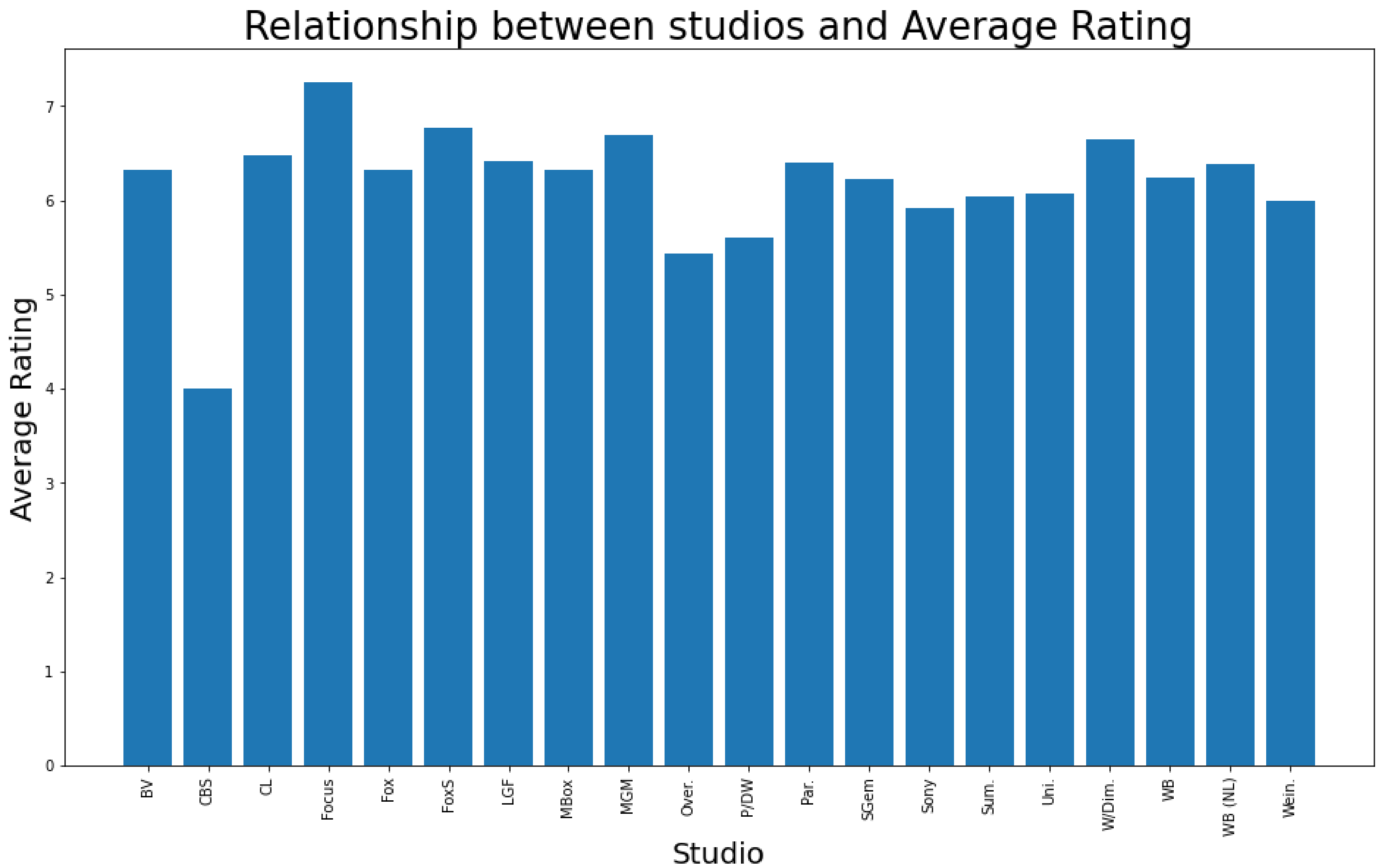
We observe the highest earning genre is `Drama`, followed by `Documentary`, `comedy`, `Horror`, `Drama/romance`, `thriller`. But we also observe that some columns contain elements of another genre, but we were unable to create a function to separate the same.

2. How does the rating of a movie affect its revenue at the box office?

To address this question, we can create a histogram plot showing the relationship between studio and average rating for movies.

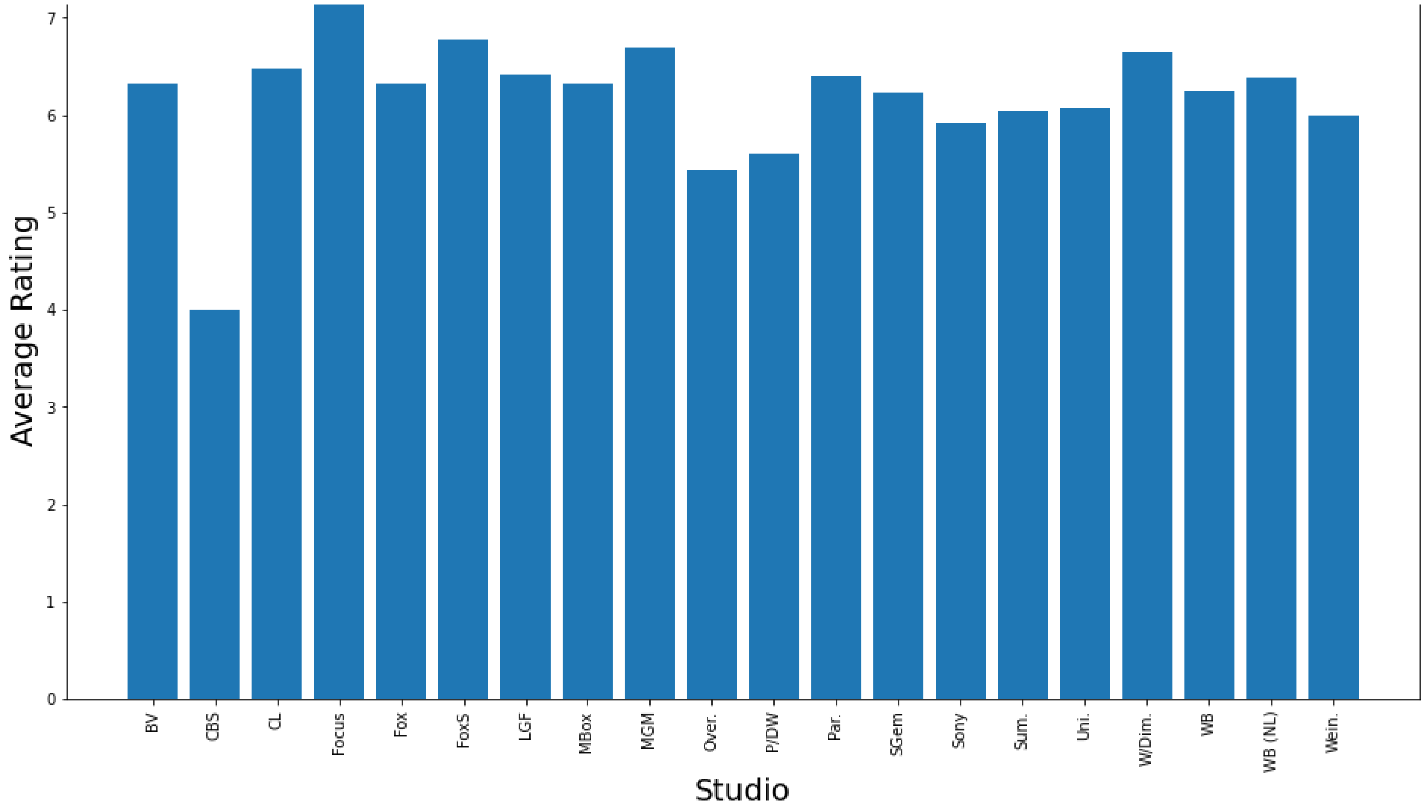
This is a plot that shows the relationship between studio and average rating for movies.We observe that the highest rated studio is Focus studio, while the Highest earning studio was WB even the second rated Fox studios did not also garner good ratings. We would need to see personal movie reviews to understand why this change is presented in our data

 # Create a bar plot of studio in relation to the average rating plt.figure(figsize=(16,9)) plt.bar(studio_data['studio'], height=studio_data['averagerating']) plt.xlabel('Studio',fontsize=20) plt.ylabel('Average Rating',fontsize=20) plt.title('Relationship between studios and Average Rating',fontsize=26) plt.xticks(rotation=90) plt.show()



We Might also need to consider which movies had the most ratings based on their duration. As seen in the Histogram below

 # Create a histogram of runtime plt.figure(figsize=(16,9)) plt.hist(studio_data['runtime_minutes'], density=True, bins=20) plt.xlabel('Runtime (minutes)',fontsize=20) plt.ylabel('Frequency',fontsize=20) plt.title('Distribution of Runtime',fontsize=26) plt.show()

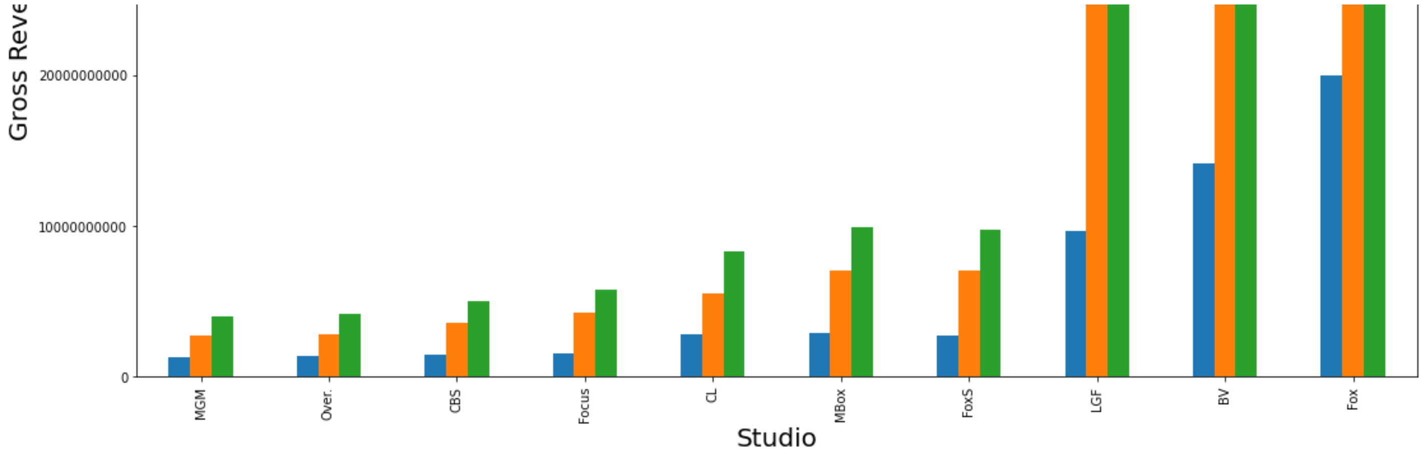


3. What are the most successful movie studios in the industry, and how do they achieve success?

To address this question, we can create a `bar chart` showing the `top 10 movie studios by revenue` with separate bars for `domestic gross`, `profit` and `worldwide gross`.

Below, we create a plot showing the earnings and expenses of the leading studios, where Warner Bros is the highest earning production studio, which can be attributed to the higher number of movies they release. We also observe that having a higher budget does not neccesarily mean that the earning will be the same as in the case between Fox and WB, where fox spent more that WB but this did not reflect on the sales.

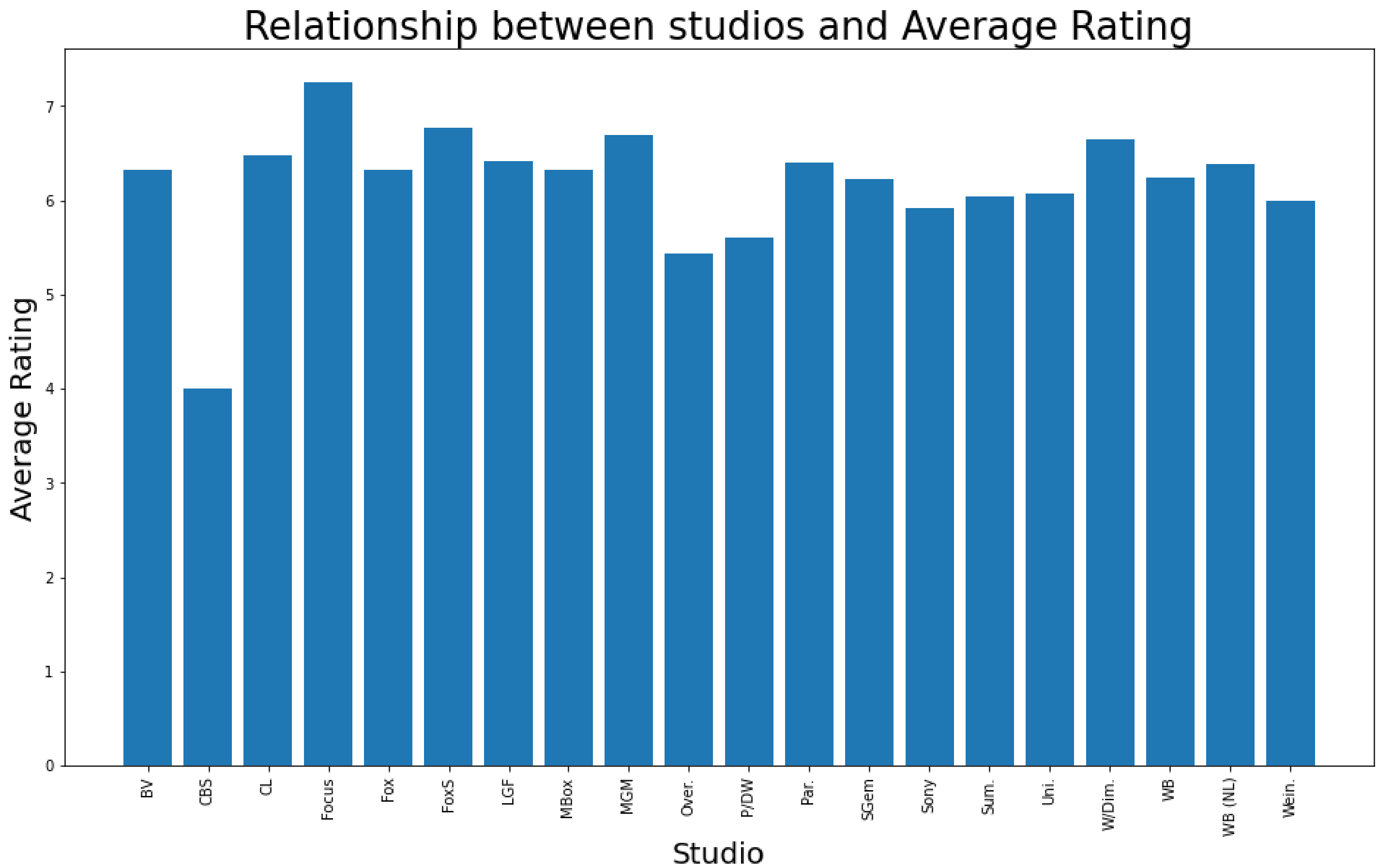
#Create a side by side bar plot of the production_budget, worldwide_gross and profit for each studio ax = grouped[['production_budget', 'profit', 'worldwide_gross']].head(10).sort_values(by='profit').plot(kind='bar', figsize=(8, 6)) # Set the title and axis labels ax.set_title('Total profit, Gross Revenue and Production Cost by Studio') ax.set_xlabel('Studio') ax.set_ylabel('Gross Revenue against profit and Production Cost (in billions)') plt.ticklabel_format(style='plain',axis='y') # Display the plot plt.show()



4. How does the budget of a movie affect its performance at the box office?

To address this question, we can create a `scatter plot` **showing the** `relationship between the budget` **of a movie and its** `domestic gross` **,** `profit` **or** `worldwide gross` **. To understand the relationship between the production budget and the gross revenue for each movie, we will create a Budget vs. Gross Revenue Scatter Plot, where each data point represents a movie title.**

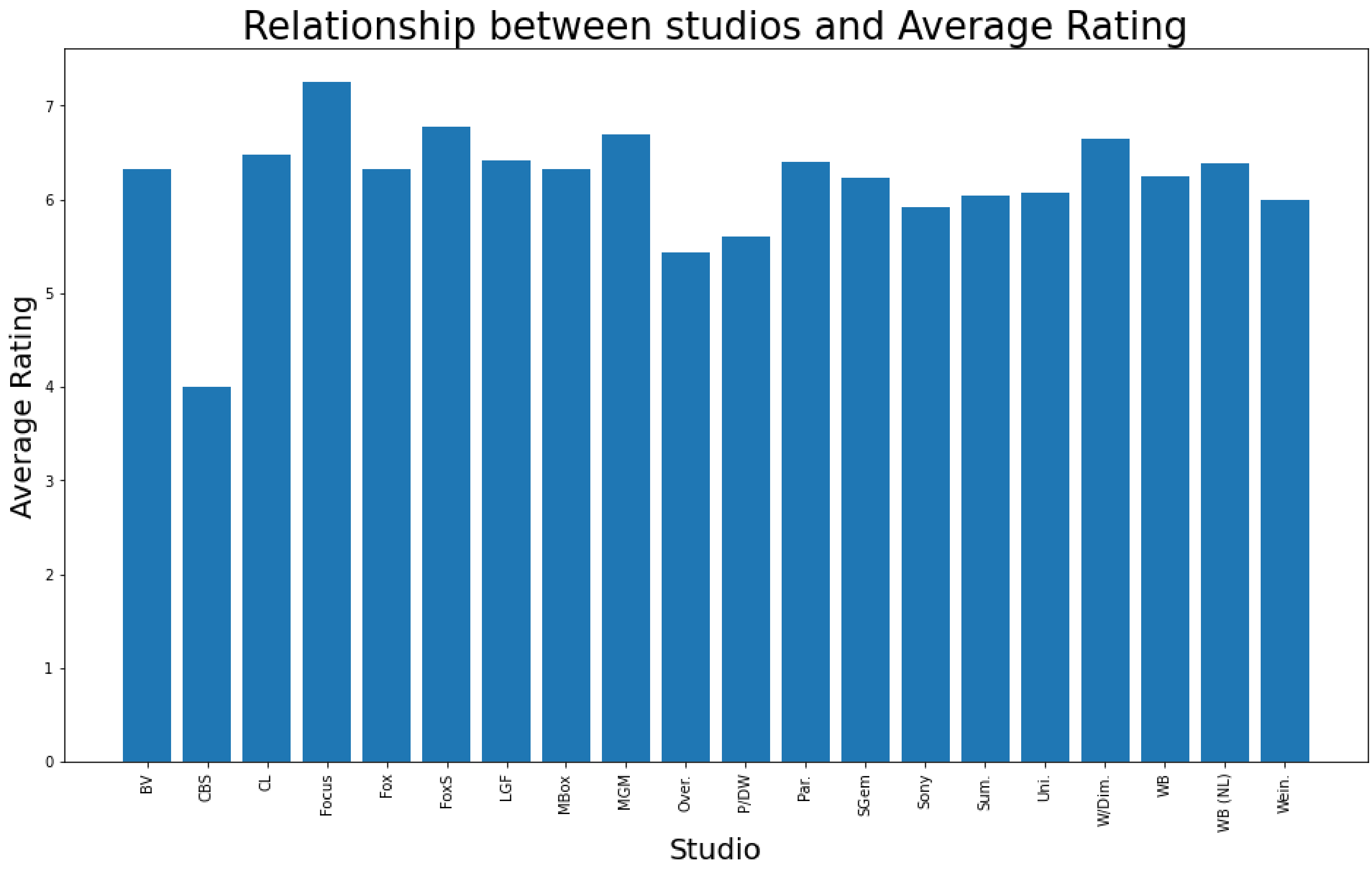
 # Creating a scatter plot of the production_budget and worldwide_gross for each movie ax = mov_analysis.plot(kind='scatter', x='production_budget', y='worldwide_gross', figsize=(28, 16)) x = mov_analysis['production_budget'] y = mov_analysis['worldwide_gross'] # Calculate the correlation coefficient corr = np.corrcoef(x, y)[0, 1] # Add a correlation line x_line = np.array([min(x), max(x)]) y_line = corr * x_line plt.plot(x_line, y_line, color='red', label=f'Correlation = {corr:.2f}') # Set the title and axis labels ax.set_title('Budget vs. Gross Revenue',fontsize=26) ax.set_xlabel('Production Budget (in millions of dollars)',fontsize=26) ax.set_ylabel('Gross Revenue (in billions of dollars)',fontsize=26) plt.ticklabel_format(style='plain',axis='y') # Display the plot plt.show()



From the plot above we observe via the corerlation gradient line, there is a positive correlation bwtween the gross revenue and production budget. The more is spent on production the higher it sells. This can probably be attributed to the genre, or studio producing the budget, which we can further investigate.

To identify the highest grossing movies and understand the studios that are the most successful, we will plot Top Grossing Movies using a bar plot

#first we will plot the top grossing movie studios and limit the results to the first ten top_movies = mov_analysis.sort_values('worldwide_gross', ascending=False).head(10) top_movies # Creating a bar plot of the worldwide_gross for each movie ax = top_movies.plot(kind='bar', x='title', y='worldwide_gross', figsize=(8, 6)) # Set the title and axis labels ax.set_title('Top Grossing Movies',fontsize=26) ax.set_xlabel('Movie Title',fontsize=16) ax.set_ylabel('Gross Revenue ',fontsize=16) plt.ticklabel_format(style='plain',axis='y')

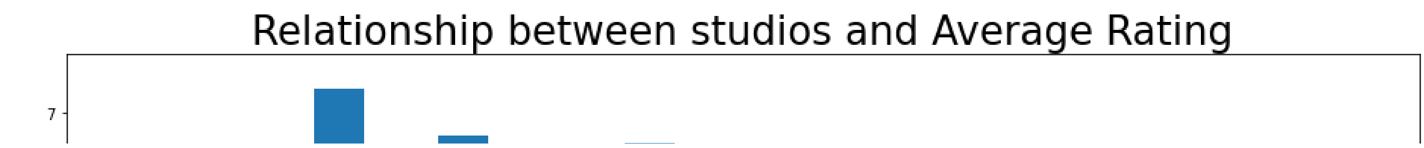


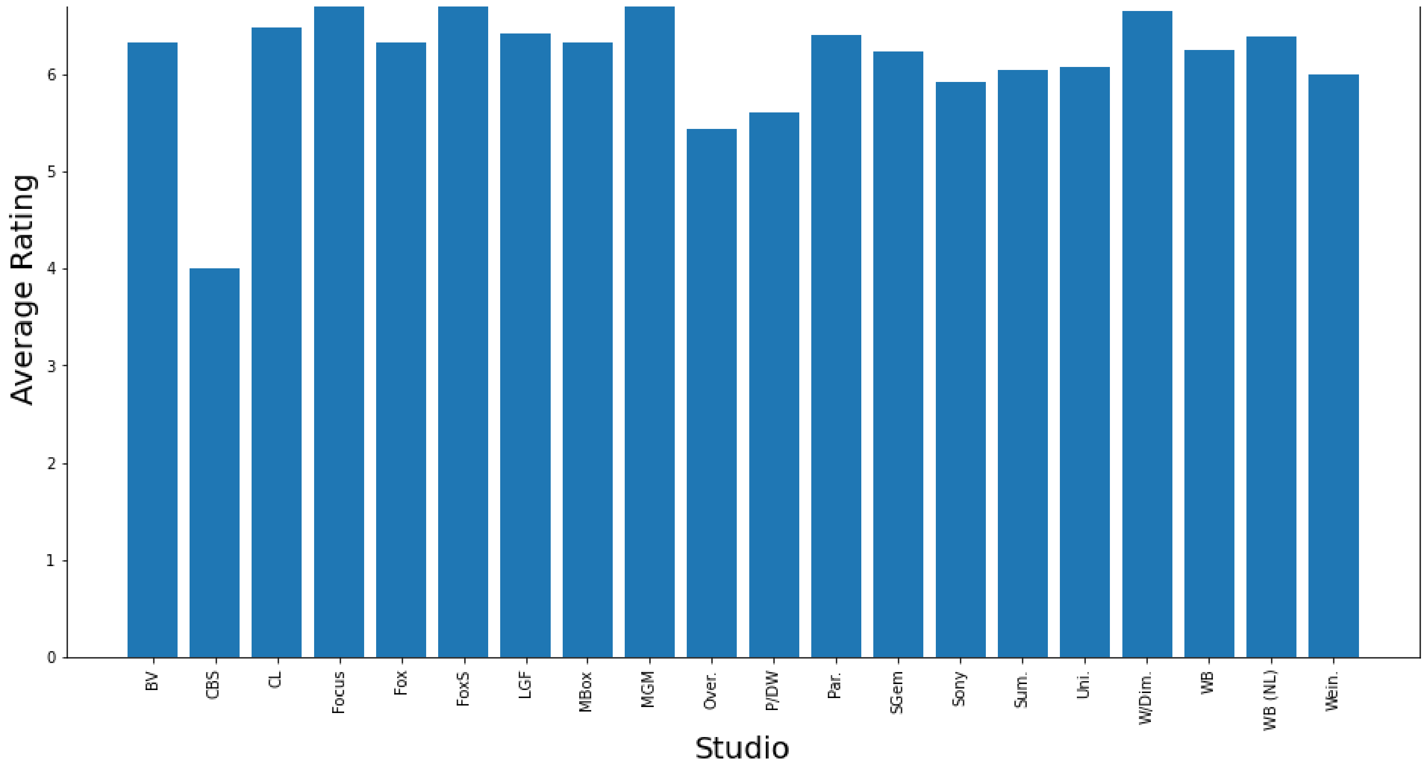
We observe that the highest grossing movies are from the genre Sci-Fi, with more than seven out of the ten coming form that genre and above 5 from the studio but it is not among the ones that are highly rated. but from our knowledge we know they do not have many productions, but the few they have make them sufficient profits.

5. Are there any specific release dates or seasons that are more profitable for movie releases?

To address this question, we can create a `line graph` showing the `average monthly revenue` for movie releases over a certain period of time, with markers for specific holiday weekends or seasons.

#we will need to import the calender funtion in python tobe able to identify holidays and caalender days #importing calender import calendar # Create a list of holiday weekends or seasons (e.g., Thanksgiving, Christmas, summer) holidays = ['Thanksgiving', 'Christmas', 'Summer'] # Create a new column to indicate whether a given month corresponds to a holiday or not grouped['holiday'] = grouped.index.astype(int).map(lambda x: any(h in calendar.month_name[x] for h in holidays)) # Create a bar plot of the monthly revenue ax = grouped.plot(kind='bar', stacked=False, figsize=(20,16)) # Create a line graphs of the foreign gross,worldwide gross and profit with markers for holidays grouped['foreign_gross'].plot(kind='line', ax=ax, color='orange') grouped['worldwide_gross'].plot(kind='line', ax=ax, color='green') grouped['profit'].plot(kind='line', ax=ax, color='red') grouped['domestic_gross'].plot(kind='line', ax=ax, color='blue') grouped['production_budget'].plot(kind='line', ax=ax, color='purple') # Add markers for holidays for idx, row in grouped[grouped['holiday']].iterrows(): ax.axvline(idx - 0.5, color='gray', linestyle='--', linewidth=3) # Set title and axis labels ax.set_title("Monthly Revenue with Markers for Holidays", fontsize=26) ax.set_xlabel("Month", fontsize=20) ax.set_ylabel('Revenue (in billions of dollars)', fontsize=20) plt.ticklabel_format(style='plain',axis='y') plt.legend(['Domestic Gross', 'Foreign Gross', 'Worldwide Gross', 'Profit', 'production_budget'], fontsize=16, loc='upper left')





From the graph above we can observe that the peach time for releasing movies was during the holiday season. Specifically during the summer months of `May, June, July` and the winter months of `November and December`. We observe that for foreign gross represents a higher contribution to the worldwide revenue, which is logically true if all countries in the world have submited revenue, but we know this is not an actual representation of the data since the foreign gross had the highest number of missing values in the dataset. The plot itself is clearly self explanatory, but the markers seem to have been hidden. it would be my suggestion to plan release of movies during the summer and winter break when many of the movie patrons have free time to attend cinemas

Conclusion

In this project, we analyzed data from five different sources to generate insights for Microsoft's new movie studio. We presented three concrete business recommendations based on our analysis:

Microsoft should focus on producing Drama, Documentary, and comedy movies. Microsoft should consider partnering with Warner Bros. and Focus studio for its new movie studio. Microsoft should consider releasing movies during the holidays. Microsoft can use these recommendations to help them decide what types of films to create.

We have observed form all the above illustrations that indeed, the movie production business is very lucrative. For microsoft to consider investing in their own studio, it will be a significant diversification of their revenue base as long as they are willing to invest the funds required. This is just a preliminary analysis and a much deeper understandind will be required to provide a detailed course of action to take. This is because we will have to take intoconsideration the whole process of movie creation from selecting the best directors, identifying script writers, looking into the relationship between actors and movie ratings. we would also need to consider viewer feedback on each movie released and its relationship to revenue and by extension production budget. Looking forward to be contracted to carry out this further analysis

Data Understanding

Reading Datasets

Data set one: Movie Gross Sales

In [33]:

```python
"""This is data on the gross sales of movies"""

#reading the data
movie_gross = pd.read_csv(".Data/bom.movie_gross.csv")

#looking into the data
#print(movie_gross)


"""We notice the data contains 3387 rows and five columns as below:"""

movie_gross
```

Out[33]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	NaN	2018
3383	Edward II (2018 re-release)	FM	4800.0	NaN	2018
3384	El Pacto	Sony	2500.0	NaN	2018
3385	The Swan	Synergetic	2400.0	NaN	2018
3386	An Actor Prepares	Grav.	1700.0	NaN	2018

3387 rows × 5 columns

Cleaning this dataset

In [34]:

```python
#next, we check wether this data contains any missing information
movie_gross.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3387 entries, 0 to 3386
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   title           3387 non-null   object 
 1   studio          3382 non-null   object 
 2   domestic_gross  3359 non-null   float64
 3   foreign_gross   2037 non-null   object 
 4   year            3387 non-null   int64  
dtypes: float64(1), int64(1), object(3)
memory usage: 132.4+ KB
```

We notice the data contains missing values in all columns except two: `title` and `year` we also notice the `foreign_gross` is an *object type yet it is an integer*;

the `Studio` column has *5* missing values, the `domestic_gross` has *28* missing values and `foreign_gross` has the highest number of missing values with `1350` rows missing values.

In [35]:

```python
movie_gross['foreign_gross'] = movie_gross['foreign_gross'].str.replace(',', '').astype(
float)
```

In [36]:

```
#counting the missing values in the other columns
movie_gross.isnull().sum()
```

Out[36]:

```
title              0
studio             5
domestic_gross    28
foreign_gross   1350
year               0
dtype: int64
```

To be able to understand the data better, we are going to subset it below to only contain data in the foreign gross without the missing rows. we will then carry out descriptive analysis of the same to understand the mean, maximum value, minimum value and the standard deviation.

Looking at the new dataset,we observe a huge unrealisting replacement on some of the values a movie could not possibly earn more in that 100% the domestic market in the foreign market; we check our previously subset data without the replaced value find if there were any occurences of the same. We observe that this is a normal occurence although not at that percentage.

This is a function that `generates a series` of `random numbers` between `min` value `600` and value `9605000` , since using the `max value` will give a skewed data set. With these numbers we fill the null values in the data, so that we can avoid removing almost 40% of the data.

In [37]:

```
def rand_gross_null(data):
    replace_null = mvg.apply(lambda x: np.random.randint(low=600, high=9605000) if pd.is
null(x) else x)
    return replace_null
```

We will test our function out below by recreating a new data set with only the Foreign gross column

In [41]:

```
#the code below drops all the missing values
mvg = movie_gross['foreign_gross'].dropna()
#we encountered a value that contained a coma(,) so we needed to remove it
mvg = mvg.replace(',', '').astype(float)
#below are the statitstical analysis of the new data frame with 2037 rows
print("Number of Rows:", mvg.count())
print("Maximum Value:", mvg.max())
print("Minimum Value:", mvg.min())
print("Mean:", mvg.mean())
print("Median Value:", mvg.median())
print("Standard Deviation:", mvg.std())
```

```
Number of Rows: 2037
Maximum Value: 960500000.0
Minimum Value: 600.0
Mean: 74872810.15046637
Median Value: 18700000.0
Standard Deviation: 137410600.84150565
```

In [42]:

```
#Creating subset
mvg = movie_gross['foreign_gross']
#testing the function
print (mvg.apply(lambda x: np.random.randint(low=600, high=9605000) if pd.isnull(x) else
x))
```

```
0        652000000.0
1        691300000.0
2        664300000.0
```

```
3        535700000.0
4        513900000.0
            ...
3382       3405435.0
3383        153671.0
3384       6352199.0
3385        478328.0
3386       1280847.0
Name: foreign_gross, Length: 3387, dtype: float64
```

Since we see it works,we will apply it below

In [43]:

```
#Here we apply the above function to our dataset
movie_gross['foreign_gross'] = rand_gross_null(movie_gross['foreign_gross'])

#confirm the changes are made.
movie_gross['foreign_gross']
```

Out[43]:

```
0        652000000.0
1        691300000.0
2        664300000.0
3        535700000.0
4        513900000.0
            ...
3382       7878625.0
3383       1782646.0
3384       5794192.0
3385       7767165.0
3386       7711530.0
Name: foreign_gross, Length: 3387, dtype: float64
```

In [44]:

```
#Glimpse into out data
movie_gross
```

Out[44]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000.0	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000.0	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000.0	2010
3	Inception	WB	292600000.0	535700000.0	2010
4	Shrek Forever After	P/DW	238700000.0	513900000.0	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	7878625.0	2018
3383	Edward II (2018 re-release)	FM	4800.0	1782646.0	2018
3384	El Pacto	Sony	2500.0	5794192.0	2018
3385	The Swan	Synergetic	2400.0	7767165.0	2018
3386	An Actor Prepares	Grav.	1700.0	7711530.0	2018

3387 rows × 5 columns

In [51]:

```
"""To be able to conduct analysis using this data
we need to remove or replace the missing values"""
#removing missing values in the studio column
movie_gross.dropna(subset=['studio'], inplace=True)
#removing missing values in the domestic_gross column
```

```
movie_gross.dropna(subset=['domestic_gross'], inplace=True)
```

In [52]:

```
#check for final changes
print(movie_gross.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3356 entries, 0 to 3386
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   title           3356 non-null   object 
 1   studio          3356 non-null   object 
 2   domestic_gross  3356 non-null   float64
 3   foreign_gross   3356 non-null   float64
 4   year            3356 non-null   int64  
dtypes: float64(2), int64(1), object(2)
memory usage: 157.3+ KB
None
```

We now see the final dataframe has a total of `3356` after final cleaning. Now our dataset is ready to use"

To be able to make a more informed decision, we will look at how such a change will affect the mean median and standard deviation for our subset dataset

In [53]:

```
#desctiptive staticsics
print("New Data Set Statistical analysis:")
print(movie_gross.describe().round())
```

```
New Data Set Statistical analysis:
       domestic_gross  foreign_gross    year
count          3356.0         3356.0  3356.0
mean       28771490.0     47207039.0  2014.0
std        67006943.0    112406645.0     2.0
min             100.0          600.0  2010.0
25%          120000.0      2748584.0  2012.0
50%         1400000.0      7039052.0  2014.0
75%        27950000.0     29700000.0  2016.0
max       936700000.0    960500000.0  2018.0
```

In the dataset we will use: The `mean` is `28771490.00`, `47293893.00`, `2014` for `domestic_gross`, `foreign_gross`, `year` columns. The `standard deviation` is `67006943.00`, `112374024.00`, `2` for `domestic_gross`, `foreign_gross`, `year` columns. The `minimum value` is `100`, `600`, `2010` for `domestic_gross`, `foreign_gross`, `year` columns. The `median` is `1400000`, `7218232.00`, `2014` for `domestic_gross`, `foreign_gross`, `year` columns. The `maximum value` is `936700000.00`, `960500000.00`, `2018` for `domestic_gross`, `foreign_gross`, `year` columns.

Data set two: Movie information

In [55]:

```
#reading additional datasets
movie_info = pd.read_table(".Data/rt.movie_info.tsv",index_col=None)

#looking into the data
movie_info
```

Out[55]:

	id	synopsis	rating	genre	director	writer	theater_date	dvd_date	curr
0	1	This gritty, fast-paced, and	R	Action and	William	Ernest Tidyman	Oct 9, 1971	Sep 25,	

	id	synopsis	rating	genre	director	writer	theater_date	dvd_date	curr
0	1	... police...	R	...Adventure|Classics|Drama	...Friedkin	Ernest Tidyman	Oct 9, 1971	... 2001	
1	3	New York City, not-too-distant-future: Eric Pa...	R	Drama|Science Fiction and Fantasy	David Cronenberg	David Cronenberg|Don DeLillo	Aug 17, 2012	Jan 1, 2013	
2	5	Illeana Douglas delivers a superb performance ...	R	Drama|Musical and Performing Arts	Allison Anders	Allison Anders	Sep 13, 1996	Apr 18, 2000	
3	6	Michael Douglas runs afoul of a treacherous su...	R	Drama|Mystery and Suspense	Barry Levinson	Paul Attanasio|Michael Crichton	Dec 9, 1994	Aug 27, 1997	
4	7	NaN	NR	Drama|Romance	Rodney Bennett	Giles Cooper	NaN	NaN	
...	...	...	...	...	...	...	...	...	
1555	1996	Forget terrorists or hijackers -- there's a ha...	R	Action and Adventure|Horror|Mystery and Suspense	NaN	NaN	Aug 18, 2006	Jan 2, 2007	
1556	1997	The popular Saturday Night Live sketch was exp...	PG	Comedy|Science Fiction and Fantasy	Steve Barron	Terry Turner|Tom Davis|Dan Aykroyd|Bonnie Turner	Jul 23, 1993	Apr 17, 2001	
1557	1998	Based on a novel by Richard Powell, when the l...	G	Classics|Comedy|Drama|Musical and Performing Arts	Gordon Douglas	NaN	Jan 1, 1962	May 11, 2004	
1558	1999	The Sandlot is a coming-of-age story about a g...	PG	Comedy|Drama|Kids and Family|Sports and Fitness	David Mickey Evans	David Mickey Evans|Robert Gunter	Apr 1, 1993	Jan 29, 2002	
1559	2000	Suspended from the force, Paris cop Hubert is ...	R	Action and Adventure|Art House and Internation...	NaN	Luc Besson	Sep 27, 2001	Feb 11, 2003	

1560 rows × 12 columns

In [56]:

```
movie_info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1560 entries, 0 to 1559
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   id            1560 non-null   int64 
 1   synopsis      1498 non-null   object
 2   rating        1557 non-null   object
 3   genre         1552 non-null   object
 4   director      1361 non-null   object
 5   writer        1111 non-null   object
 6   theater_date  1201 non-null   object
 7   dvd_date      1201 non-null   object
 8   currency      340 non-null    object
```

```
 9   box_office    340 non-null    object
 10  runtime       1530 non-null   object
 11  studio        494 non-null    object
dtypes: int64(1), object(11)
memory usage: 146.4+ KB
```

In [57]:

```
#counting the missing values
movie_info.isnull().sum()
```

Out[57]:

```
id                 0
synopsis          62
rating             3
genre              8
director         199
writer           449
theater_date     359
dvd_date         359
currency        1220
box_office      1220
runtime           30
studio          1066
dtype: int64
```

We observe that only the id column does not have missing values looking at the extreem number of missing valuesin the currency and box office columns, we have no option but to drop them. we can keep the studio column and try later to fill them from movie_gross dataset. alternatively we can discard it from use at this time.

In [58]:

```
movie_info = movie_info.dropna()
movie_info
```

Out[58]:

	id	synopsis	rating	genre	director	writer	theater_date	dvd_date	currency
1	3	New York City, not-too-distant-future: Eric Pa...	R	Drama|Science Fiction and Fantasy	David Cronenberg	David Cronenberg|Don DeLillo	Aug 17, 2012	Jan 1, 2013	$
6	10	Some cast and crew from NBC's highly acclaimed...	PG-13	Comedy	Jake Kasdan	Mike White	Jan 11, 2002	Jun 18, 2002	$
7	13	Stewart Kane, an Irishman living in the Austra...	R	Drama	Ray Lawrence	Raymond Carver|Beatrix Christian	Apr 27, 2006	Oct 2, 2007	$
15	22	Two-time Academy Award Winner Kevin Spacey giv...	R	Comedy|Drama|Mystery and Suspense	George Hickenlooper	Norman Snider	Dec 17, 2010	Apr 5, 2011	$
18	25	From ancient Japan's most enduring tale, the e...	PG-13	Action and Adventure|Drama|Science Fiction and...	Carl Erik Rinsch	Chris Morgan|Hossein Amini	Dec 25, 2013	Apr 1, 2014	$
...	...	...	...	...	...	...	...	...	...
		This holiday							

	id	synopsis	rating	genre	director	writer	theater_date	dvd_date	currency
1530	1968	This holiday acclaimed filmmaker Camer...	PG	Comedy|Drama	Cameron Crowe	Aline Brosh McKenna|Cameron Crowe	Dec 23, 2011	Apr 3, 2012	$
1537	1976	Embrace of the Serpent features the encounter,...	NR	Action and Adventure|Art House and International	Ciro Guerra	Ciro Guerra|Jacques Toulemonde Vidal	Feb 17, 2016	Jun 21, 2016	$
1541	1980	A band of renegades on the run in outer space ...	PG-13	Action and Adventure|Science Fiction and Fantasy	Joss Whedon	Joss Whedon	Sep 30, 2005	Dec 20, 2005	$
1542	1981	Money, Fame and the Knowledge of English. In I...	NR	Comedy|Drama	Gauri Shinde	Gauri Shinde	Oct 5, 2012	Nov 20, 2012	$
1545	1985	A woman who joins the undead against her will ...	R	Horror|Mystery and Suspense	Sebastian Gutierrez	Sebastian Gutierrez	Jun 1, 2007	Oct 9, 2007	$

235 rows × 12 columns

The above function significantly reduces out dataset, therefore using in in the analysis will mean a reduced number of datapoints therefore we will not have a comprehensive analysis. We can opt to use it in the end while sampling to try and explain results of our findings or to try and get customer feedbackin case needed

Data set three: Movie Reviews

In [59]:

```
#reading additional datasets
movie_reviews = pd.read_table(".Data/rt.reviews.tsv",encoding='unicode_escape',index_col=
False)

#looking into the data
movie_reviews
```

Out[59]:

	id	review	rating	fresh	critic	top_critic	publisher	date
0	3	A distinctly gallows take on contemporary fina...	3/5	fresh	PJ Nabarro	0	Patrick Nabarro	November 10, 2018
1	3	It's an allegory in search of a meaning that n...	NaN	rotten	Annalee Newitz	0	io9.com	May 23, 2018
2	3	... life lived in a bubble in financial dealin...	NaN	fresh	Sean Axmaker	0	Stream on Demand	January 4, 2018
3	3	Continuing along a line introduced in last yea...	NaN	fresh	Daniel Kasman	0	MUBI	November 16, 2017
4	3	... a perverse twist on neorealism...	NaN	fresh	NaN	0	Cinema Scope	October 12, 2017
...	...	...	...	...	...	...	...	...
54427	2000	The real charm of this trifle is the deadpan c...	NaN	fresh	Laura Sinagra	1	Village Voice	September 24, 2002
54428	2000	NaN	1/5	rotten	Michael Szymanski	0	Zap2it.com	September 21, 2005
54429	2000	NaN	2/5	rotten	Emanuel Levy	0	EmanuelLevy.Com	July 17, 2005

	id	review	rating	fresh	critic	top_critic	publisher	date
54430	2000	NaN	2.5/5	rotten	Christopher Null	0	Filmcritic.com	September 7, 2003
54431	2000	NaN	3/5	fresh	Nicolas Lacroix	0	Showbizz.net	November 12, 2002

54432 rows × 8 columns

In [60]:

```
#counting the missing values
print("Number of missing values per column:")

print(movie_reviews.isnull().sum())

print("""We observe the data contain a very high number of mising values
therefore we might consider not using this dataset at this time,
just as the above dataset.
although the initial data contains 54432 rows,
and the resulting rows after the drop are 33988
""")
```

```
Number of missing values per column:
id                0
review         5563
rating        13517
fresh             0
critic         2722
top_critic        0
publisher       309
date              0
dtype: int64
We observe the data contain a very high number of mising values
therefore we might consider not using this dataset at this time,
just as the above dataset.
although the initial data contains 54432 rows,
and the resulting rows after the drop are 33988
```

In [61]:

```
movie_reviews = movie_reviews.dropna()
movie_reviews
```

Out[61]:

	id	review	rating	fresh	critic	top_critic	publisher	date
0	3	A distinctly gallows take on contemporary fina...	3/5	fresh	PJ Nabarro	0	Patrick Nabarro	November 10, 2018
6	3	Quickly grows repetitive and tiresome, meander...	C	rotten	Eric D. Snider	0	EricDSnider.com	July 17, 2013
7	3	Cronenberg is not a director to be daunted by ...	2/5	rotten	Matt Kelemen	0	Las Vegas CityLife	April 21, 2013
11	3	While not one of Cronenberg's stronger films, ...	B-	fresh	Emanuel Levy	0	EmanuelLevy.Com	February 3, 2013
12	3	Robert Pattinson works mighty hard to make Cos...	2/4	rotten	Christian Toto	0	Big Hollywood	January 15, 2013
...	...	...	...	...	...	...	...	...
54419	2000	Sleek, shallow, but frequently amusing.	2.5/4	fresh	Gene Seymour	1	Newsday	September 27, 2002
54420	2000	The spaniel-eyed Jean Reno infuses Hubert with...	3/4	fresh	Megan Turner	1	New York Post	September 27, 2002
54421	2000	Manages to be somewhat well-acted, not badly a...	1.5/4	rotten	Bob Strauss	0	Los Angeles Daily News	September 27, 2002
54422	2000	Arguably the best script that Besson has	3.5/5	fresh	Wade	0	Boxoffice	September

	id	review	rating	fresh	critic	top_critic	publisher	date
		writt...			Major		Magazine	27, 2002
54424	2000	Dawdles and drags when it should pop; it doesn...	1.5/5	rotten	Manohla Dargis	1	Los Angeles Times	September 26, 2002

33988 rows × 8 columns

Data set four: Movies

In [62]:

```
#reading additional datasets
movies = pd.read_csv(".Data/.tmdb.movies.csv",index_col=0)#,encoding='unicode_escape',ind
ex_col=False)

#looking into the data
movies
```

Out[62]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_
0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	7.7	
1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	
2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	
3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	
4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	
...	...	...	...	...	...	...	...	...	
26512	[27, 18]	488143	en	Laboratory Conditions	0.600	2018-10-13	Laboratory Conditions	0.0	
26513	[18, 53]	485975	en	_EXHIBIT_84xxx_	0.600	2018-05-01	_EXHIBIT_84xxx_	0.0	
26514	[14, 28, 12]	381231	en	The Last One	0.600	2018-10-01	The Last One	0.0	
26515	[10751, 12, 28]	366854	en	Trailer Made	0.600	2018-06-22	Trailer Made	0.0	
26516	[53, 27]	309885	en	The Church	0.600	2018-10-05	The Church	0.0	

26517 rows × 9 columns

In [63]:

```
#counting the missing values
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26517 entries, 0 to 26516
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   genre_ids          26517 non-null  object
 1   id                 26517 non-null  int64
 2   original_language  26517 non-null  object
 3   original_title     26517 non-null  object
 4   popularity         26517 non-null  float64
 5   release_date       26517 non-null  object
 6   title              26517 non-null  object
 7   vote_average       26517 non-null  float64
 8   vote_count         26517 non-null  int64
```

```
8   vote_count         26517 non-null  int64
dtypes: float64(2), int64(2), object(5)
memory usage: 2.0+ MB
```

From this dataframe, we observe there are no missing values, but upon investigation of the data we notice that they used `placeholders` in the data such as `- _EXHIBIT_84xxx_` in the column `original title`, `0.600` for the `popularity column` and `0.0` in the `vote_average column`, among other unidentified values. we can assume that this will be a separate value in the dataset representing the unknown, or anonymous records.

Data set five: Movie Budgets

In [69]:

```
#reading additional datasets
movie_budgets = pd.read_csv(".Data/.tn.movie_budgets.csv",index_col=0)#,encoding='unicode
_escape',index_col=False)

#looking into the data
movie_budgets
```

Out[69]:

id	release_date	movie	production_budget	domestic_gross	worldwide_gross
1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279
2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875
3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350
4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963
5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747
...	...	...	...	...	...
78	Dec 31, 2018	Red 11	$7,000	$0	$0
79	Apr 2, 1999	Following	$6,000	$48,482	$240,495
80	Jul 13, 2005	Return to the Land of Wonders	$5,000	$1,338	$1,338
81	Sep 29, 2015	A Plague So Pleasant	$1,400	$0	$0
82	Aug 5, 2005	My Date With Drew	$1,100	$181,041	$181,041

5782 rows × 5 columns

Here we will write a function that will strip `commas` and `dollar sign` on the `production_budget`, `domestic_gross` and `worldwide_gross`.

In [71]:

```
#striping from production budget column
movie_budgets.production_budget = movie_budgets.production_budget.str.replace('$', '', r
egex=True)
movie_budgets.production_budget = movie_budgets.production_budget.str.replace(',', '', r
egex=True)
movie_budgets.production_budget = movie_budgets.production_budget.astype(int)
```

In [70]:

```
#striping from domestic gross column
movie_budgets.domestic_gross = movie_budgets.domestic_gross.str.replace('$', '',regex=Tru
e)
movie_budgets.domestic_gross = movie_budgets.domestic_gross.str.replace(',', '',regex=Tru
e)
movie_budgets.domestic_gross = movie_budgets.domestic_gross.astype(int)
```

In [72]:

```
#striping from domestic gross column
movie_budgets.worldwide_gross = movie_budgets.worldwide_gross.str.replace('$', '', regex=
True)
movie_budgets.worldwide_gross = movie_budgets.worldwide_gross.str.replace(',', '', regex=
True)
movie_budgets = movie_budgets.rename(columns={'movie':'title'})
movie_budgets.worldwide_gross = movie_budgets.worldwide_gross.astype(float)
```

In [73]:

```
#checking if changes are effected in the dataset
movie_budgets.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5782 entries, 1 to 82
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   release_date       5782 non-null   object
 1   title              5782 non-null   object
 2   production_budget  5782 non-null   int32
 3   domestic_gross     5782 non-null   int32
 4   worldwide_gross    5782 non-null   float64
dtypes: float64(1), int32(2), object(2)
memory usage: 225.9+ KB
```

We see that the data now has the columns as `integers and float(decimal)` data types from the previous `object` type with the initial data

In [74]:

```
#reconfirming the data has no null values
movie_budgets.isnull().sum()
```

Out[74]:

```
release_date         0
title                0
production_budget    0
domestic_gross       0
worldwide_gross      0
dtype: int64
```

Joining the relevant Datasets

Below we will now join our three datasets, `movies`, `movie_bedgets` and `movie_gross`

In [75]:

```
#joining the data sets
mov_analysis = pd.concat([movie_budgets,movie_gross,movies],names=['title'],join='outer'
)
mov_analysis.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 35655 entries, 1 to 26516
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   release_date       32299 non-null  object
 1   title              35655 non-null  object
 2   production_budget  5782 non-null   float64
 3   domestic_gross     9138 non-null   float64
 4   worldwide_gross    5782 non-null   float64
 5   studio             3356 non-null   object
 6   foreign_gross      3356 non-null   float64
 7   year               3356 non-null   float64
```

```
 8   genre_ids            26517 non-null  object
 9   id                   26517 non-null  float64
 10  original_language    26517 non-null  object
 11  original_title       26517 non-null  object
 12  popularity           26517 non-null  float64
 13  vote_average         26517 non-null  float64
 14  vote_count           26517 non-null  float64
dtypes: float64(9), object(6)
memory usage: 4.4+ MB
```

We observe we have a total of 35655 rows. with a majority of the columns having missing values and studio having the least. From here we will drop all other rows and remain only with those in the studio column

In [76]:

```
#This codedrops all the unwanted columns
mov_analysis.drop(['genre_ids', 'id','original_language','original_title','popularity','
vote_average','vote_count'], axis=1, inplace=True)
```

In [77]:

```
mov_analysis.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 35655 entries, 1 to 26516
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   release_date       32299 non-null  object
 1   title              35655 non-null  object
 2   production_budget  5782 non-null   float64
 3   domestic_gross     9138 non-null   float64
 4   worldwide_gross    5782 non-null   float64
 5   studio             3356 non-null   object
 6   foreign_gross      3356 non-null   float64
 7   year               3356 non-null   float64
dtypes: float64(5), object(3)
memory usage: 2.4+ MB
```

Futher cleaning is expected to be done on this data. We then fill the existing missing values using standard practice,

#This checked to see if we had duplicate values which we deal with below duplicated_movs = mov_analysis[mov_analysis.duplicated() == True] duplicated_movs

In [78]:

```
mov_analysis.isnull().sum()
```

Out[78]:

```
release_date          3356
title                    0
production_budget    29873
domestic_gross       26517
worldwide_gross      29873
studio               32299
foreign_gross        32299
year                 32299
dtype: int64
```

In [79]:

```
mov_analysis = mov_analysis.drop_duplicates()
mov_analysis
```

Out[79]:

	release_date	title	production_budget	domestic_gross	worldwide_gross	studio	foreign_gross	year
1	Dec 18, 2009	Avatar	425000000.0	760507625.0	2.776345e+09	NaN	NaN	NaN

	release_date	title	production_budget	domestic_gross	worldwide_gross	studio	foreign_gross	year
2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09	NaN	NaN	NaN
3	Jun 7, 2019	Dark Phoenix	350000000.0	42762350.0	1.497624e+08	NaN	NaN	NaN
4	May 1, 2015	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09	NaN	NaN	NaN
5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000.0	620181382.0	1.316722e+09	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...
26512	2018-10-13	Laboratory Conditions	NaN	NaN	NaN	NaN	NaN	NaN
26513	2018-05-01	_EXHIBIT_84xxx_	NaN	NaN	NaN	NaN	NaN	NaN
26514	2018-10-01	The Last One	NaN	NaN	NaN	NaN	NaN	NaN
26515	2018-06-22	Trailer Made	NaN	NaN	NaN	NaN	NaN	NaN
26516	2018-10-05	The Church	NaN	NaN	NaN	NaN	NaN	NaN

34628 rows × 8 columns

In [80]:

```
mov_analysis.dropna(subset=['production_budget'], inplace=True)

mov_analysis
```

Out[80]:

	release_date	title	production_budget	domestic_gross	worldwide_gross	studio	foreign_gross	year
1	Dec 18, 2009	Avatar	425000000.0	760507625.0	2.776345e+09	NaN	NaN	NaN
2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09	NaN	NaN	NaN
3	Jun 7, 2019	Dark Phoenix	350000000.0	42762350.0	1.497624e+08	NaN	NaN	NaN
4	May 1, 2015	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09	NaN	NaN	NaN
5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000.0	620181382.0	1.316722e+09	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...
78	Dec 31, 2018	Red 11	7000.0	0.0	0.000000e+00	NaN	NaN	NaN
79	Apr 2, 1999	Following	6000.0	48482.0	2.404950e+05	NaN	NaN	NaN
80	Jul 13, 2005	Return to the Land of Wonders	5000.0	1338.0	1.338000e+03	NaN	NaN	NaN
81	Sep 29, 2015	A Plague So Pleasant	1400.0	0.0	0.000000e+00	NaN	NaN	NaN
82	Aug 5, 2005	My Date With Drew	1100.0	181041.0	1.810410e+05	NaN	NaN	NaN

5782 rows × 8 columns

Here we calculate the `foreign gross` by `subtracting` `domestic gross` from `worldwide gross`

In [81]:

```
#this is a function that calculates the foreign gross
def calculate_foreign_gross(row):
    return row['worldwide_gross'] - row['domestic_gross']

#applying the function to our data set
mov_analysis['foreign_gross'] = mov_analysis.apply(calculate_foreign_gross, axis=1)
```

Here we will strip the year from the release date column, which we have used to fill our dataset year column as

below ten we will create a new column for the month

In [82]:

```
#Filling the year column
mov_analysis['year'] = pd.to_datetime(mov_analysis['release_date']).dt.strftime('%Y')

#creating the month column
mov_analysis['month'] = pd.to_datetime(mov_analysis['release_date']).dt.strftime('%m')
```

For the `Studio column`, we will first `create a list of studios` from the `movie_gross dataset` above, then we will use this `list` to fill the `values in our mov_analysis` data set

In [83]:

```
#creating the list
studios = movie_gross.studio
```

In [84]:

```
#populating our dataset
mov_analysis.studio.fillna(studios, inplace=True)

#confirming changes
mov_analysis.studio.isnull().sum()
```

Out[84]:

```
0
```

In [85]:

```
#confirming changes to entire dataset
mov_analysis.isnull().sum()
```

Out[85]:

```
release_date         0
title                0
production_budget    0
domestic_gross       0
worldwide_gross      0
studio               0
foreign_gross        0
year                 0
month                0
dtype: int64
```

Now our dataframe is ready for analysis, although for the `foreign_gross` and `worldwide_gross` values are still written in `scientific format` but we need them to be in `plain format`. Below is a function that changes from the default scientific display format

In [86]:

```
#here we write the function to remove the scientific format
pd.options.display.float_format = '{:.2f}'.format

#checking if the dataset for changes
mov_analysis.head()
```

Out[86]:

	release_date	title	production_budget	domestic_gross	worldwide_gross	studio	foreign_gross	year	month
1	Dec 18, 2009	Avatar	425000000.00	760507625.00	2776345279.00	BV	2015837654.00	2009	12
2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000.00	241063875.00	1045663875.00	WB	804600000.00	2011	05
3	Jun 7, 2019	Dark Phoenix	350000000.00	42762350.00	149762350.00	WB	107000000.00	2019	06

	release_date	title	production_budget	domestic_gross	worldwide_gross	studio	foreign_gross	year	month
4	May 1, 2015	Avengers: Age of Ultron	330600000.00	459005868.00	1403013963.00	P/DW	944008095.00	2015	05
5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000.00	620181382.00	1316721747.00	Sum.	696540365.00	2017	12

We might need a column denoting the profit which is calculated by subtracting the production budget from worldwide gross

In [87]:

```
#creating a profit column
def calculate_profit(row):
    return (row['domestic_gross']+row['foreign_gross']) - row['production_budget']
mov_analysis['profit'] = mov_analysis.apply(calculate_profit, axis=1)
```

We observe that some of the columns are negative values indicative of a loss which we do not need in our analysis at the moment. Below we will remove these values from ur dataframe

In [88]:

```
#cpnverting the loss values to NaN
mov_analysis['profit'] = mov_analysis['profit'].mask(mov_analysis['profit'] < 0)

# drop rows containing NaNs
mov_analysis = mov_analysis.dropna(subset=['profit'])

#checking the changes have been applied
mov_analysis['profit']
```

Out[88]:

```
1     2351345279.00
2      635063875.00
4     1072413963.00
5      999721747.00
6     1747311220.00
           ...
74       2034928.00
75        834926.00
76         64644.00
79        234495.00
82        179941.00
Name: profit, Length: 3657, dtype: float64
```

Data Visualization

Understanding Project Requirements

The project must focus on identifying the types of movies that are currently doing well in the market and provide actionable insights to the head of Microsoft's new movie studio. The stakeholders in this project include Microsoft executives, investors, and potential moviegoers who are interested in watching new and exciting movies. The problem is to find the right strategy for Microsoft's new movie studio by understanding the audience's preferences and current market trends. By identifying the types of films that are currently successful, Microsoft can create a plan that aligns with the market's demand and attract potential customers.

To explore what types of films are currently doing the best at the box office, we can analyze the provided data set of movies. The relevant columns to consider for this analysis are `genres`, `averagerating`, `domestic_gross`, and `worldwide_gross`.

In [89]:

```
#descriptive analysis of the data
```

```
#descriptive analysis of the data
mov_analysis.describe()
```

Out[89]:

	production_budget	domestic_gross	worldwide_gross	foreign_gross	profit
count	3657.00	3657.00	3657.00	3657.00	3657.00
mean	38447721.79	62587298.34	138943577.51	76356279.17	100495855.72
std	47453605.88	78270587.17	204789444.00	134994772.79	170733189.29
min	1100.00	0.00	71644.00	0.00	349.00
25%	7000000.00	14439985.00	25480031.00	2567207.00	11134983.00
50%	20000000.00	38122000.00	67348218.00	27395664.00	37977250.00
75%	50000000.00	79366978.00	164675402.00	87022647.00	112886353.00
max	425000000.00	936662225.00	2776345279.00	2015837654.00	2351345279.00

Subsetting by `Studio` and the the `Sum` of their `Budget`, `Domestic`, `Worldwide`, `Foreign` and `Profit` columns we will name this data subset `Studio Summary`

In [90]:

```
#grouping by Studio and the the Sum of their Budget, Domestic, Worldwide, Foreign and Pro
fit columns
#we will name this data subset Studio Summary
studio_summary = mov_analysis.groupby('studio').agg({
    'production_budget': 'sum',
    'domestic_gross': 'sum',
    'worldwide_gross': 'sum',
    'foreign_gross': 'sum',
    'profit': 'sum'
})
studio_summary.reset_index()
```

Out[90]:

	studio	production_budget	domestic_gross	worldwide_gross	foreign_gross	profit
0	BV	14173634000.00	23959847912.00	52879877062.00	28920029150.00	38706243062.00
1	CBS	1435010000.00	2055813774.00	4973767583.00	2917953809.00	3538757583.00
2	CL	2799071082.00	4065241035.00	8340880826.00	4275639791.00	5541809744.00
3	Focus	1509900000.00	2685566653.00	5761086729.00	3075520076.00	4251186729.00
4	Fox	19980829000.00	31233473293.00	70045929302.00	38812456009.00	50065100302.00
5	FoxS	2704238783.00	4698429938.00	9787922596.00	5089492658.00	7083683813.00
6	LGF	9698155000.00	16632679630.00	35671185399.00	19038505769.00	25973030399.00
7	MBox	2913972000.00	4871408848.00	9948721717.00	5077312869.00	7034749717.00
8	MGM	1312525000.00	2075133257.00	4004886259.00	1929753002.00	2692361259.00
9	Over.	1372014000.00	1868979028.00	4155404195.00	2286425167.00	2783390195.00
10	P/DW	5615096000.00	8585075582.00	19187197855.00	10602122273.00	13572101855.00
11	Par.	11063581610.00	18116862224.00	40495961418.00	22379099194.00	29432379808.00
12	SGem	8338516774.00	13595249886.00	30618606519.00	17023356633.00	22280089745.00
13	Sony	11408368000.00	19417126888.00	44110498153.00	24693371265.00	32702130153.00
14	Sum.	6981870785.00	12211700033.00	26497390962.00	14285690929.00	19515520177.00
15	Uni.	12681402841.00	20488539604.00	46101719962.00	25613180358.00	33420317121.00
16	W/Dim.	1499747000.00	2586753596.00	5369647769.00	2782894173.00	3869900769.00
17	WB	19611643704.00	31947148973.00	72802231312.00	40855082339.00	53190587608.00
18	WB (NL)	4130533000.00	6271094715.00	14405148580.00	8134053865.00	10274615580.00

	studio	production_budget	domestic_gross	worldwide_gross	foreign_gross	profit
19	Weln.	1373210000.00	1515625147.00	2958598739.00	1442973592.00	1585388739.00

Descriptive statistical information as below for the `studio_summary` data subset

In [91]:

```
#we can get some Descriptive statistical information as below:
studio_summary.describe()
```

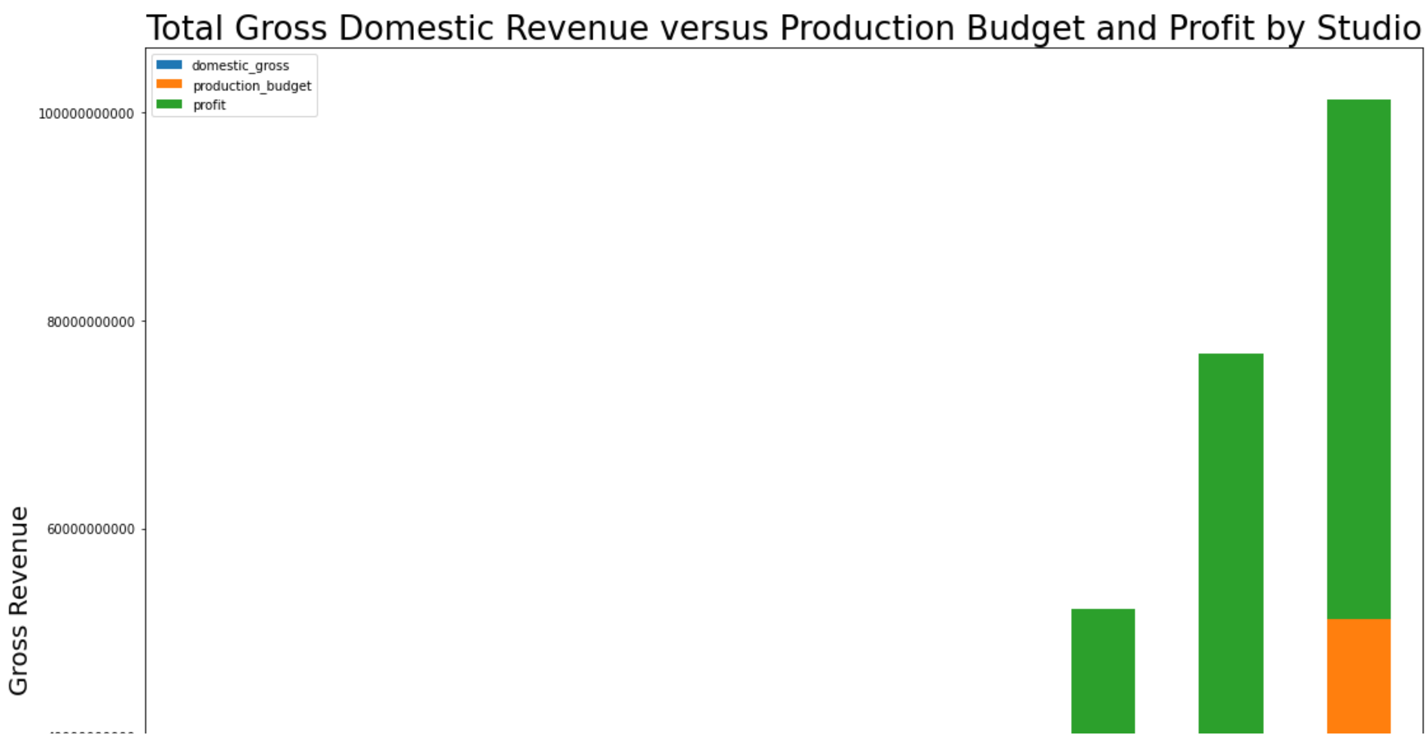
Out[91]:

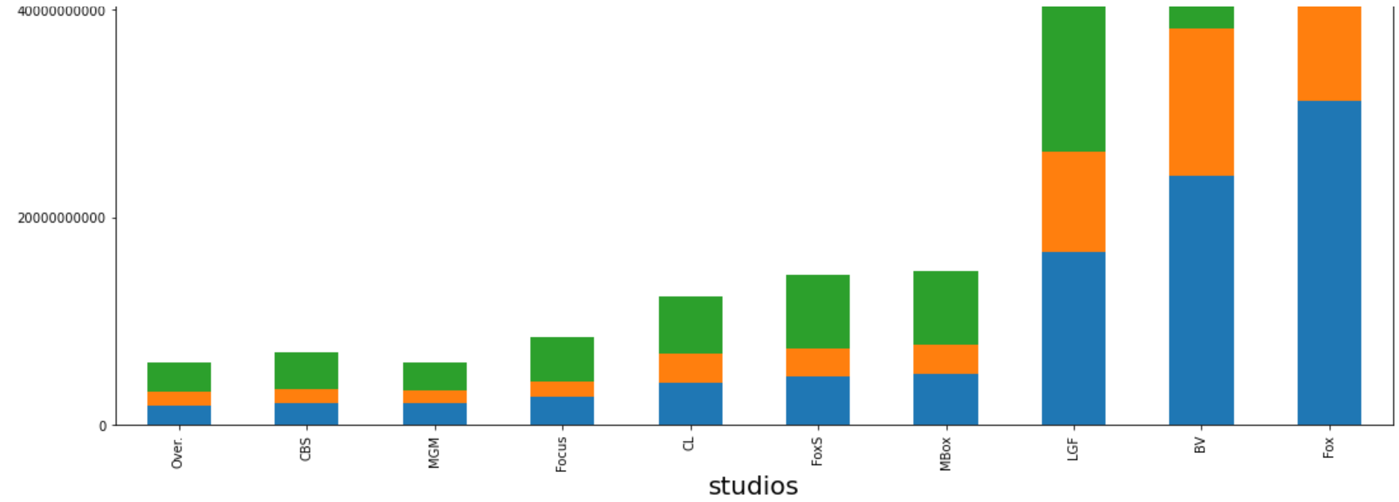
	production_budget	domestic_gross	worldwide_gross	foreign_gross	profit
count	20.00	20.00	20.00	20.00	20.00
mean	7030165928.95	11444087500.80	25405833146.85	13961745646.05	18375667217.90
std	6096976358.84	9973909315.30	22578112570.14	12610668762.11	16492933108.64
min	1312525000.00	1515625147.00	2958598739.00	1442973592.00	1585388739.00
25%	1507361750.00	2660863388.75	5663226989.00	3036128509.25	4155865239.00
50%	4872814500.00	7428085148.50	16796173217.50	9368088069.00	11923358717.50
75%	11149778207.50	18441928390.00	41399595601.75	22957667211.75	30249817394.25
max	19980829000.00	31947148973.00	72802231312.00	40855082339.00	53190587608.00

We can create a stacked `Bar Graph` to Show the relashionship between the `domestic_gross`, `production_budget` and `profit` to be able to undertand which are the higest grossing movie studios

In [92]:

```
#Create a stacked bar plot of the domestic_gross, production_budget and profit for high d
omestic revenue studios
ax = studio_summary[['domestic_gross', 'production_budget', 'profit']].head(10).sort_val
ues(by='domestic_gross').plot(kind='bar', stacked=True, figsize=(18, 16))

# Set the title and axis labels
ax.set_title('Total Gross Domestic Revenue versus Production Budget and Profit by Studio'
, fontsize=26)
ax.set_xlabel('studios', fontsize=20)
ax.set_ylabel('Gross Revenue', fontsize=20)
plt.ticklabel_format(style='plain',axis='y')
# Display the plot
plt.show()
```





We observe the highest earning Movie studio by Domesticgross revenue is Fox, followed by BV, which is an acronym for Walt Disney, Third is Lionsgate.

note: Buena Vista is a brand name that has historically been used for divisions and subsidiaries of The Walt Disney Company, whose primary studios, the Walt Disney Studios, are located on Buena Vista Street in Burbank, California.

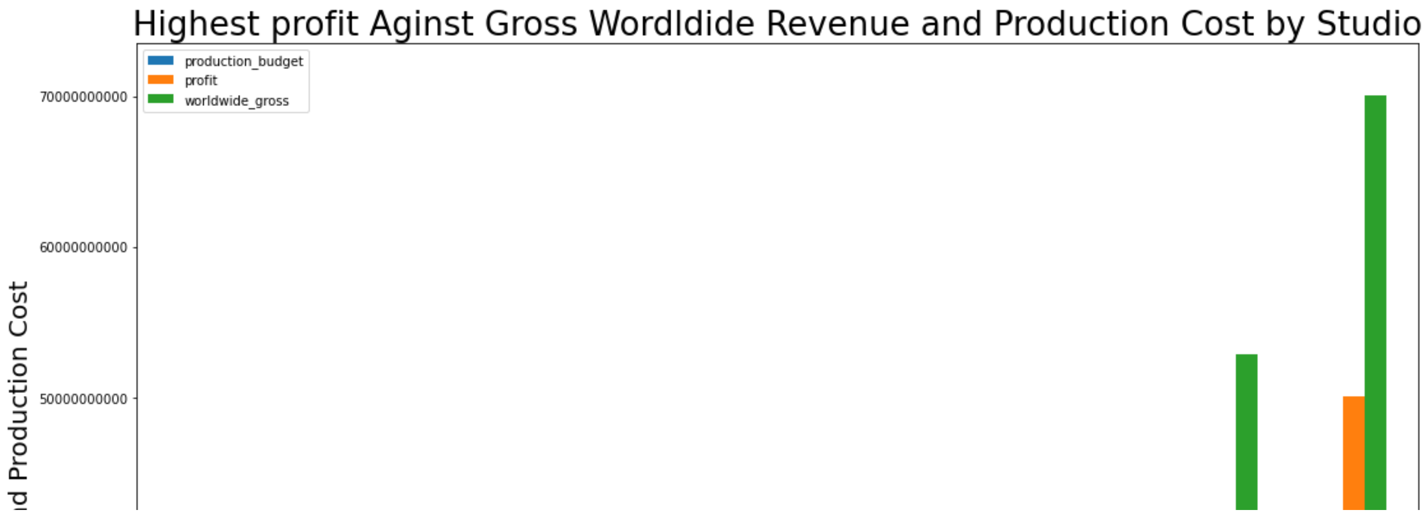
3. What are the most successful movie studios in the industry, and how do they achieve success?

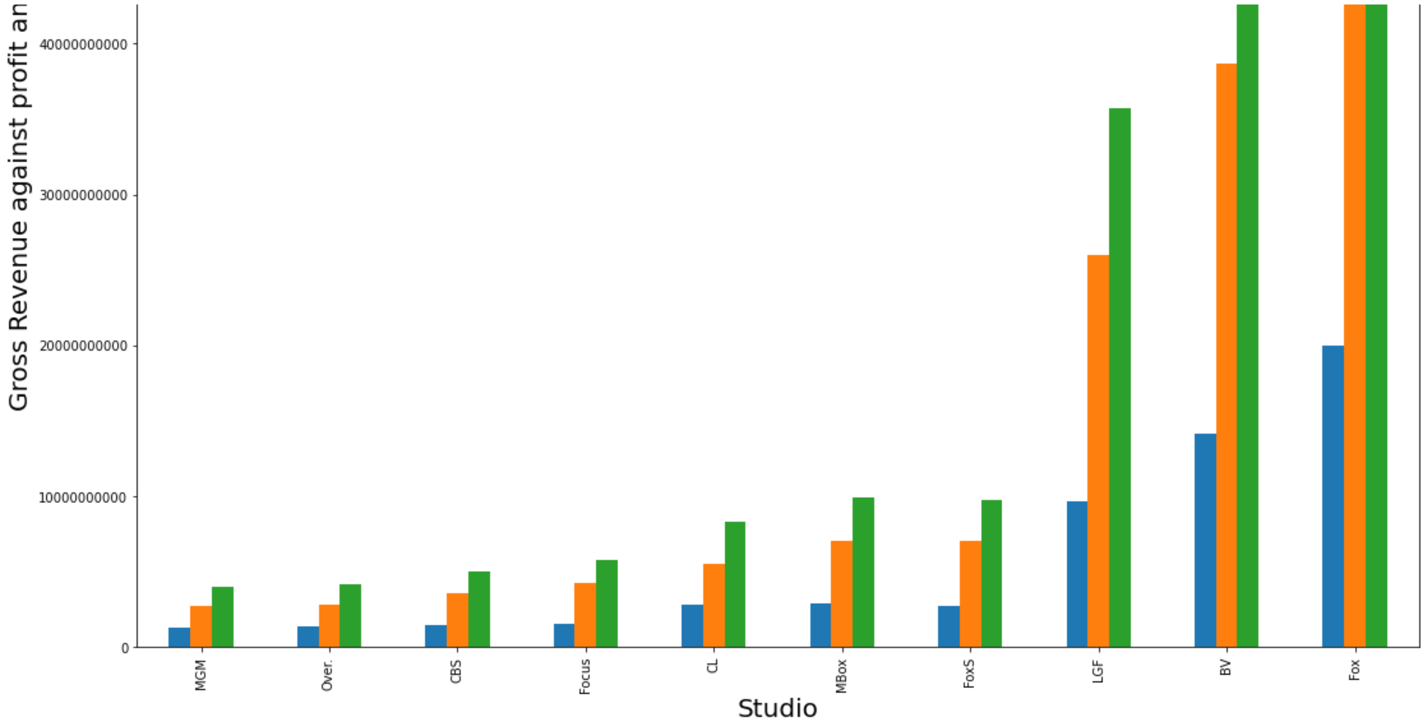
To address this question, we can create a `bar chart` showing the `top 10 movie studios by revenue` with separate bars for `domestic gross`, `profit` and `worldwide gross`.

Below, we create a plot showing the earnings and expenses of the leading studios, where Warner Bros is the highest earning production studio, which can be attributed to the higher number of movies they release. We also observe that having a higher budget does not neccesarily mean that the earning will be the same as in the case between `Fox` and `WB`, where fox spent more that WB but this did not reflect on the sales.

In [93]:

```
#Create a side by side bar plot of the production_budget, worldwide_gross and profit for each studio
ax = studio_summary[['production_budget', 'profit', 'worldwide_gross']].head(10).sort_values(by='profit').plot(kind='bar', figsize=(18, 16))

# Set the title and axis labels
ax.set_title('Highest profit Aginst Gross Wordldide Revenue and Production Cost by Studio', fontsize=26)
ax.set_xlabel('Studio', fontsize=20)
ax.set_ylabel('Gross Revenue against profit and Production Cost', fontsize=20)
plt.ticklabel_format(style='plain',axis='y')

# Display the plot
plt.show()
```





4. How does the budget of a movie affect its performance at the box office?

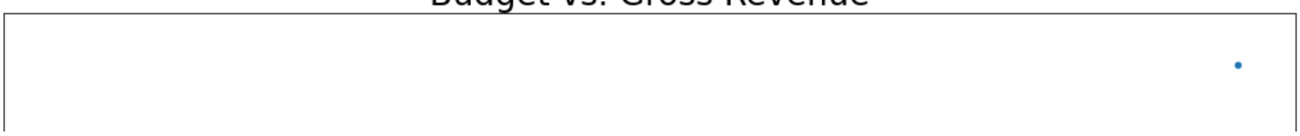
To understand the relationship between the production budget and the gross revenue for each movie, we will create a Budget vs. Gross Revenue Scatter Plot, where each data point represents a movie title.

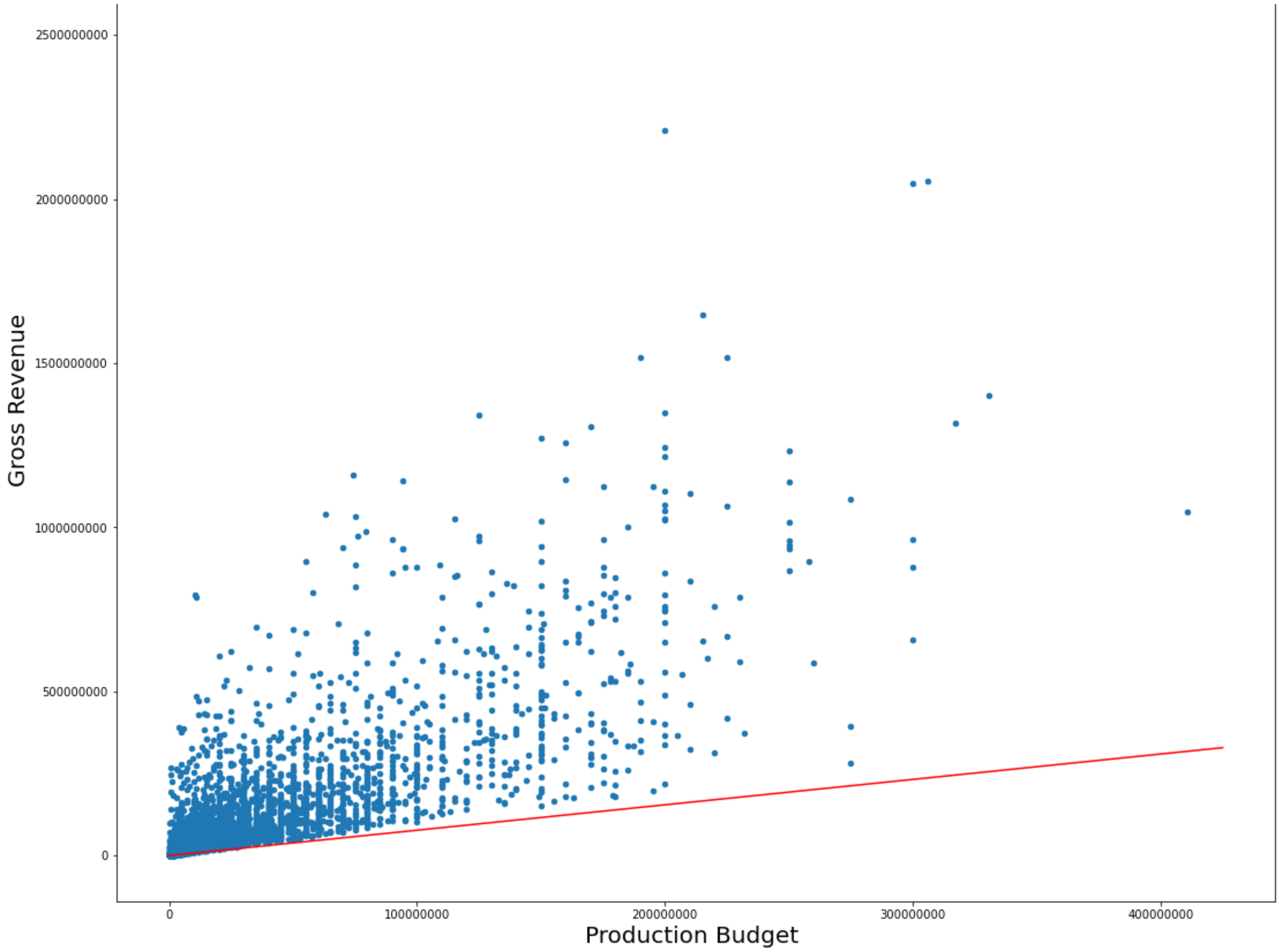
from matplotlib import ticker fig = plt.figure(figsize=(16, 10), dpi=80) # Creates one subplot within our figure and uses the classes fig and ax fig, ax = plt.subplots(figsize=(16, 10), dpi= 80, facecolor='w', edgecolor='k') # Uses hue to add an extra element, and changes the palette chart = sns.scatterplot(x='production_budget', y='worldwide_gross', data = mov_analysis, hue=mov_analysis.studio.values, legend='full', alpha = .7, palette="BrBG") # Setting axis ticks and formulating numbers ax.yaxis.set_major_locator(ticker.MaxNLocator(nbins=10)) ax.yaxis.set_minor_locator(ticker.MaxNLocator(nbins=10)) ax.yaxis.set_major_formatter(ticker.StrMethodFormatter('${x:,.0f}')) ax.xaxis.set_major_locator(ticker.MultipleLocator(500)) ax.xaxis.set_minor_locator(ticker.MultipleLocator(100)) # Naming the visual and each axis fig.suptitle("Budget vs. Gross Revenue", fontsize=26) ax.set_xlabel("Production Budget ") ax.set_ylabel("Gross Revenue") # Creating the legend ax.get_legend().set_title("Studio") plt.tight_layout()

In [94]:

```
# Creating a scatter plot of the production_budget and worldwide_gross for each movie
ax = mov_analysis.plot(kind='scatter', x='production_budget', y='worldwide_gross', figsi
ze=(18, 16))

x = mov_analysis['production_budget']
y = mov_analysis['worldwide_gross']

# Calculate the correlation coefficient
corr = np.corrcoef(x, y)[0, 1]
# Add a correlation line
x_line = np.array([min(x), max(x)])
y_line = corr * x_line
plt.plot(x_line, y_line, color='red', label=f'Correlation = {corr:.2f}')

# Set the title and axis labels
ax.set_title('Budget vs. Gross Revenue',fontsize=26)
ax.set_xlabel('Production Budget ',fontsize=20)
ax.set_ylabel('Gross Revenue',fontsize=20)
plt.ticklabel_format(style='plain',axis='both')
# Display the plot
plt.show()
```

Budget vs. Gross Revenue



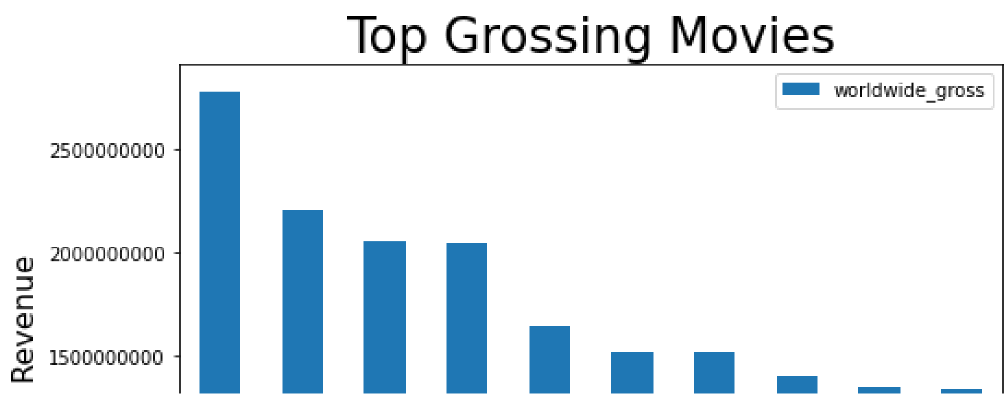


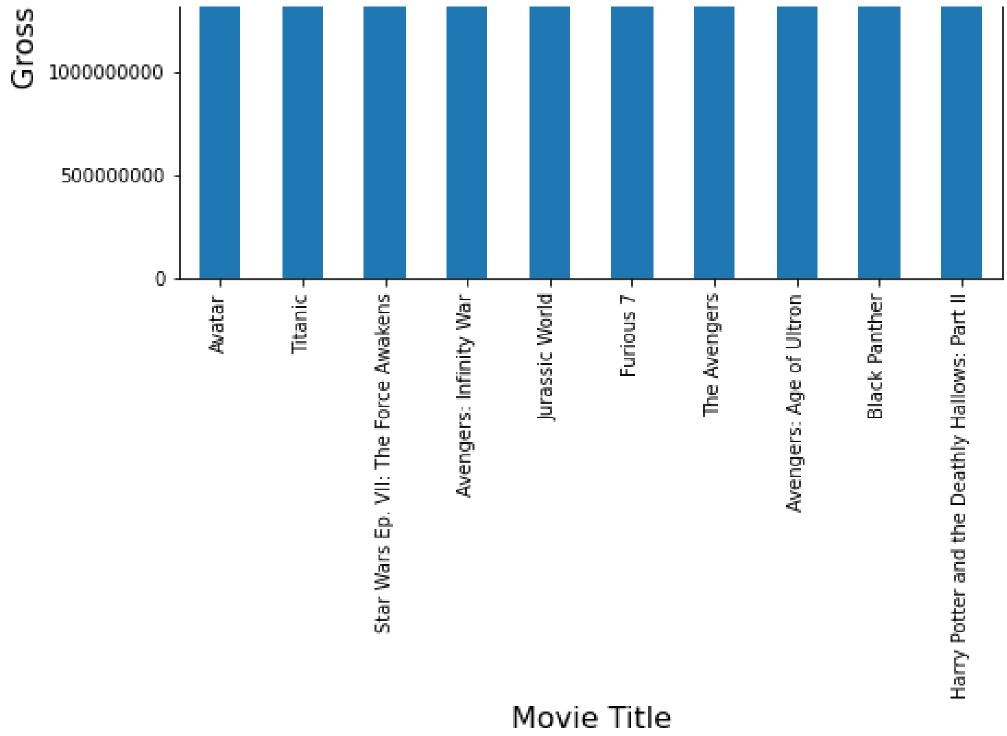
From the plot above we observe via the corerlation gradient line, there is a positive correlation bwtween the gross revenue and production budget. The more is spent on production the higher it sells. This can probably be attributed to the genre, or studio producing the budget, which we can further investigate.

To identify the highest grossing movies and understand the studios that are the most successful, we will plot Top Grossing Movies using a bar plot

In [95]:

```
#first we will plot the top grossing movie studios and limitthe results to the first ten
top_movies = mov_analysis.sort_values('worldwide_gross', ascending=False).head(10)
top_movies

# Creating a bar plot of the worldwide_gross for each movie
ax = top_movies.plot(kind='bar', x='title', y='worldwide_gross', figsize=(8, 6))

# Set the title and axis labels
ax.set_title('Top Grossing Movies',fontsize=26)
ax.set_xlabel('Movie Title',fontsize=16)
ax.set_ylabel('Gross Revenue',fontsize=16)
plt.ticklabel_format(style='plain',axis='y')
```





We observe that the highest grossing movies are from the genre Sci-Fi, with more than seven out of the ten coming form that genre and above 5 from the studio but it is not among the ones that are highly rated. but from our knowledge we know they do not have many productions, but the few they have make then sufficient profits.

To help us understand the `breakdown of revenue by year`, we will Group the data by `year` and `sum` up the `worldwide_gross` and `foreign_gross` and `profit` for each year this will help us identify the years which have highest overall revenue. We can plot a Revenue Breakdwon by year and name this data `Year Summary`. The data as grouped above creates a dataset with all income values sumed. Since we are only intrested on the highest earning we will convert those below `1 billion` to `NaN values`, then remove them from our data.

In [96]:

```
# Group the data by year and sum up the worldwide_gross and foreign_gross and profit for each year
year_summary = mov_analysis.groupby('year').agg({
    'domestic_gross': 'sum',
    'foreign_gross': 'sum',
    'profit': 'sum'
})
# replace negative values with NaNs
year_summary = year_summary.mask(year_summary < 1000000000)

# drop rows containing NaNs
year_summary = year_summary.dropna()
year_summary.reset_index()
```

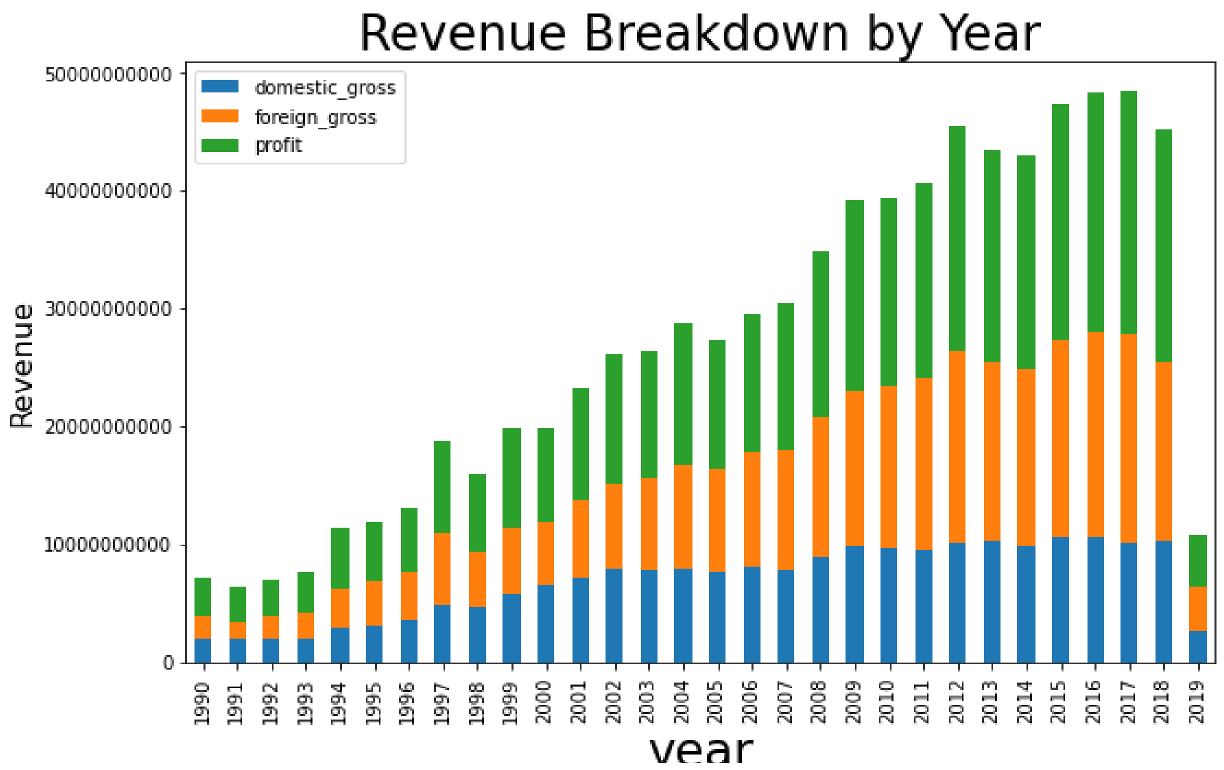
Out[96]:

	year	domestic_gross	foreign_gross	profit
0	1990	2139610766.00	1830464541.00	3296945307.00
1	1991	2044263126.00	1474080369.00	2887835495.00
2	1992	2143190979.00	1817696086.00	3187287065.00
3	1993	2131425430.00	2138381732.00	3512500162.00
4	1994	2989373953.00	3307578572.00	5097925525.00
5	1995	3102538199.00	3818473834.00	4987762033.00
6	1996	3654105944.00	4075651409.00	5521693743.00
7	1997	4946239094.00	6019675245.00	7792227339.00

	year	domestic_gross	foreign_gross	profit
8	1998	[illegible]	[illegible]	[illegible]
9	1999	5852663894.00	5650623645.00	8453348539.00
10	2000	6634811651.00	5362310638.00	7917284245.00
11	2001	7240203401.00	6513482594.00	9632515995.00
12	2002	7993400656.00	7202169910.00	10928942947.00
13	2003	7831848685.00	7775048483.00	10804647168.00
14	2004	8027563855.00	8786255249.00	12017434129.00
15	2005	7706144478.00	8718082719.00	11029893015.00
16	2006	8242055967.00	9650646271.00	11752615238.00
17	2007	7790485187.00	10275542188.00	12492197375.00
18	2008	8925829470.00	11837090477.00	14098323057.00
19	2009	9903984609.00	13091652650.00	16234187259.00
20	2010	9717675504.00	13801822713.00	15976002567.00
21	2011	9506919561.00	14565340239.00	16562694800.00
22	2012	10158011181.00	16304880269.00	19143501450.00
23	2013	10339098615.00	15138993013.00	18056491628.00
24	2014	9915273301.00	15029534898.00	18108688199.00
25	2015	10644473187.00	16688175760.00	20114723947.00
26	2016	10687910128.00	17279694136.00	20454754264.00
27	2017	10158212850.00	17697352239.00	20702299089.00
28	2018	10289249318.00	15252585878.00	19632935196.00
29	2019	2742165888.00	3649695133.00	4387561021.00

In [97]:

```
#creating a stacked bar plot of the grouped data above
ax = year_summary.plot(kind='bar', stacked=True, figsize=(10,6))

#setting title and axis labels
ax.set_title("Revenue Breakdown by Year",fontsize=26)
ax.set_xlabel("year",fontsize=26)
ax.set_ylabel('Revenue',fontsize=16)
plt.ticklabel_format(style='plain',axis='y')
```



We will repeat the above steps but now the subset will be by `month`. Group the data by `month` and `sum` up the `worldwide_gros`s and `foreign_gross`, `domestic_gross`, `production_budget` and `profit`.

In [98]:

```
# Group the data by month and sum up the worldwide_gross and foreign_gross, domestic_gros
s, production_budget
#and profit
monthly = mov_analysis.groupby('month').agg({
    'domestic_gross': 'sum',
    'foreign_gross': 'sum',
    'profit': 'sum',
    'worldwide_gross': 'sum',
    'production_budget': 'sum'
})

# replace negative values with NaNs
monthly = monthly.mask(monthly < 1000000000)

# drop rows containing NaNs
monthly = monthly.dropna()
monthly.reset_index()
```

Out[98]:

	month	domestic_gross	foreign_gross	profit	worldwide_gross	production_budget
0	01	7550323878.00	7399922407.00	10496416285.00	14950246285.00	4453830000.00
1	02	12767877786.00	13580160427.00	18734621966.00	26348038213.00	7613416247.00
2	03	17155434138.00	19175549066.00	24883332356.00	36330983204.00	11447650848.00
3	04	11368308258.00	14375176673.00	18091836931.00	25743484931.00	7651648000.00
4	05	26485869779.00	38486884837.00	47950845020.00	64972754616.00	17021909596.00
5	06	30375291513.00	36052707190.00	49181570765.00	66427998703.00	17246427938.00
6	07	25754194156.00	34643219718.00	44293617899.00	60397413874.00	16103795975.00
7	08	14845559073.00	13871989765.00	19590353119.00	28717548838.00	9127195719.00
8	09	10088019025.00	10885521338.00	14366041823.00	20973540363.00	6607498540.00
9	10	12740378645.00	13713606394.00	18778800039.00	26453985039.00	7675185000.00
10	11	27180397386.00	36937405425.00	47006555095.00	64117802811.00	17111247716.00
11	12	32570096379.00	40112769681.00	54139353060.00	72682866060.00	18543513000.00

We will recreate the plot above but using the `month` instead.

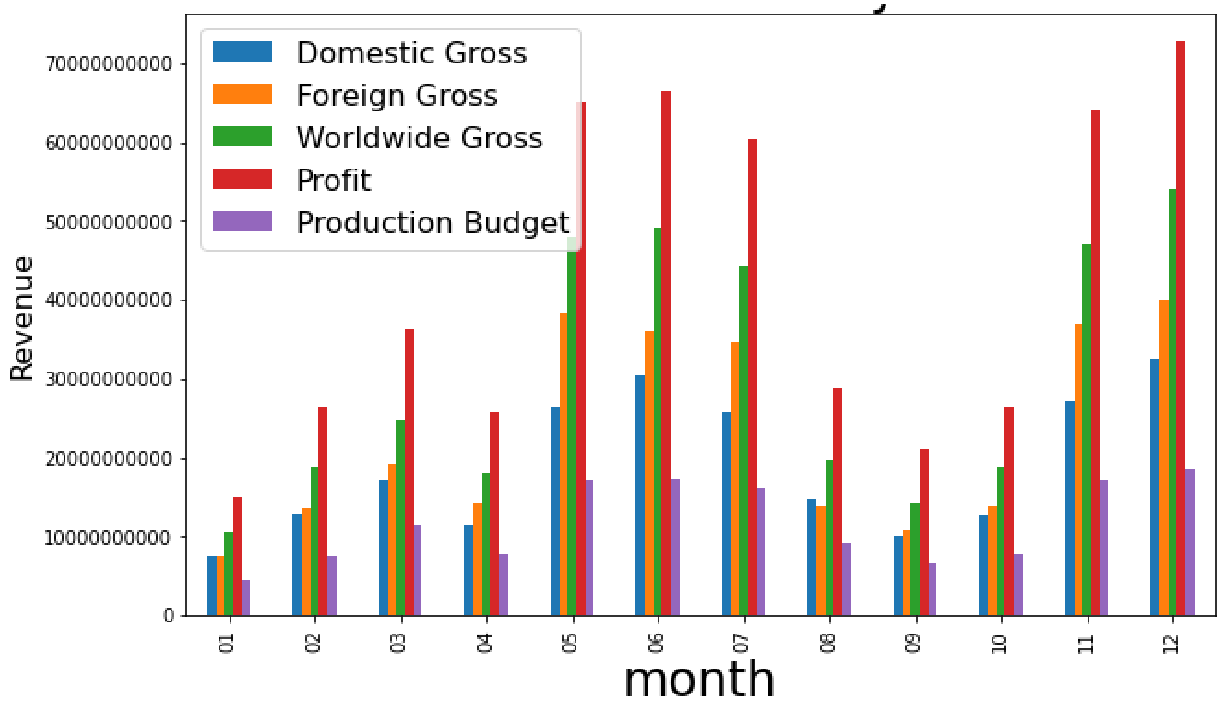
In [99]:

```
#creating a stacked bar plot of the grouped data above
ax = monthly.plot(kind='bar', stacked=False, figsize=(10,6))

#setting title and axis labels
ax.set_title("Revenue Breakdown by month",fontsize=26)
ax.set_xlabel("month",fontsize=26)
ax.set_ylabel('Revenue',fontsize=16)
plt.ticklabel_format(style='plain',axis='y')
plt.legend(['Domestic Gross', 'Foreign Gross', 'Worldwide Gross', 'Profit','Production Bu
dget'], fontsize=16, loc='upper left')
```

Out[99]:

```
<matplotlib.legend.Legend at 0x240db1799d0>
```

Revenue Breakdown by month



Although the graph above showed us months we need to convert it to a line plot in order to answer our question five. below is the code that changes the above bar chart to a line chart

In [100]:

```
category = monthly[['domestic_gross', 'foreign_gross', 'profit', 'worldwide_gross', 'pro
duction_budget']]
category.reset_index()
```

Out[100]:

	month	domestic_gross	foreign_gross	profit	worldwide_gross	production_budget
0	01	7550323878.00	7399922407.00	10496416285.00	14950246285.00	4453830000.00
1	02	12767877786.00	13580160427.00	18734621966.00	26348038213.00	7613416247.00
2	03	17155434138.00	19175549066.00	24883332356.00	36330983204.00	11447650848.00
3	04	11368308258.00	14375176673.00	18091836931.00	25743484931.00	7651648000.00
4	05	26485869779.00	38486884837.00	47950845020.00	64972754616.00	17021909596.00
5	06	30375291513.00	36052707190.00	49181570765.00	66427998703.00	17246427938.00
6	07	25754194156.00	34643219718.00	44293617899.00	60397413874.00	16103795975.00
7	08	14845559073.00	13871989765.00	19590353119.00	28717548838.00	9127195719.00
8	09	10088019025.00	10885521338.00	14366041823.00	20973540363.00	6607498540.00
9	10	12740378645.00	13713606394.00	18778800039.00	26453985039.00	7675185000.00
10	11	27180397386.00	36937405425.00	47006555095.00	64117802811.00	17111247716.00
11	12	32570096379.00	40112769681.00	54139353060.00	72682866060.00	18543513000.00

In [101]:

```
#we will need to import the calender funtion in python tobe able to identify holidays and
calender days

#importing calender
import calendar

# Create a list of holiday weekends or seasons (e.g., Thanksgiving, Christmas, summer)
holidays = ['Thanksgiving', 'Christmas', 'Summer']

# Create a new column to indicate whether a given month corresponds to a holiday or not
monthly['holiday'] = monthly.index.astype(int).map(lambda x: any(h in calendar.month_nam
```

```
e[x] for h in holidays))

# Create a bar plot of the monthly revenue
ax = monthly.plot(kind='bar', stacked=False, figsize=(20,16))

# Create a line graphs of the foreign gross,worldwide gross and profit with markers for h
olidays
monthly['foreign_gross'].plot(kind='line', ax=ax, color='orange')
monthly['worldwide_gross'].plot(kind='line', ax=ax, color='green')
monthly['profit'].plot(kind='line', ax=ax, color='red')
monthly['domestic_gross'].plot(kind='line', ax=ax, color='blue')
monthly['production_budget'].plot(kind='line', ax=ax, color='purple')

# Add markers for holidays
for idx, row in monthly[monthly['holiday']].iterrows():
    ax.axvline(idx - 0.5, color='gray', linestyle='--', linewidth=3)

# Set title and axis labels
ax.set_title("Monthly Revenue with Markers for Holidays", fontsize=26)
ax.set_xlabel("Month", fontsize=20)
ax.set_ylabel('Revenue (in billions of dollars)', fontsize=20)
plt.ticklabel_format(style='plain',axis='y')
plt.legend(['Domestic Gross', 'Foreign Gross', 'Worldwide Gross', 'Profit', 'production_b
udget'], fontsize=16, loc='upper left')
```

Out[101]:

```
<matplotlib.legend.Legend at 0x240d6b30700>
```

From the graph above we can observe that the peach time for releasing movies was during the holiday season. Specifically during the summer months of `May, June, July` and the winter months of `November and December`. We observe that for foreign gross represents a higher contribution to the worldwide revenue, which is logically true if all countries in the world have submited revenue, but we know this is not an actual representation of the data since the foreign gross had the highest number of missing values in the dataset. The

plot itself is clearly self explanatory, but the markers seem to have been hidden. it would be my suggestion to plan release of movies during the summer and winter break when many of the movie patrons have free time to attend cinemas

Reading Final dataset

Im Database

Reading Databases is a bit different abd more complex than reading a tabular dataset or csv. It is shown below.

In [102]:

```
# Open up a connection
conn = sqlite3.connect('.Data/.im.db')
# Initialize a cursor
cursor = conn.cursor()
```

A database is a collection of different tables linked together via keys, which appear on every table creating a connection

In [103]:

```
#Here we will look at all the tables present in the database
table_name_query = """SELECT name
                      AS 'Table Names'
                      FROM sqlite_master
                      WHERE type='table';"""

pd.read_sql(table_name_query, conn)
```

Out[103]:

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

Below we will look at each table independently, but only for the tables relevant toour analysis at this time

table one Movie Basics

In [104]:

```
#table one Movie Basics
imdb_mov_basic = pd.read_sql("""
SELECT *
FROM movie_basics

""",conn)
imdb_mov_basic
```

Out[104]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.00	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.00	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.00	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.00	Comedy,Drama,Fantasy
...	...	...	...	...	...	...
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.00	Drama
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Documentary
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Comedy
146142	tt9916730	6 Gunn	6 Gunn	2017	116.00	None
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentary

146144 rows × 6 columns

table two Movie Basics

In [105]:

```
#table two Movie Ratings
imdb_mov_ratings = pd.read_sql("""
SELECT *
FROM movie_ratings

""",conn)
imdb_mov_ratings
```

Out[105]:

	movie_id	averagerating	numvotes
0	tt10356526	8.30	31
1	tt10384606	8.90	559
2	tt1042974	6.40	20
3	tt1043726	4.20	50352
4	tt1060240	6.50	21
...	...	...	...
73851	tt9805820	8.10	25
73852	tt9844256	7.50	24
73853	tt9851050	4.70	14
73854	tt9886934	7.00	5
73855	tt9894098	6.30	128

73856 rows × 3 columns

table three Movie Basics

In [106]:

```
#table Three Movie Akas
imdb_mov_aka = pd.read_sql("""
SELECT *
```

```
FROM movie_akas

""",conn)
imdb_mov_aka
```

Out[106]:

	movie_id	ordering	title	region	language	types	attributes	is_original_title
0	tt0369610	10	Джурасик свят	BG	bg	None	None	0.00
1	tt0369610	11	Jurashikku warudo	JP	None	imdbDisplay	None	0.00
2	tt0369610	12	Jurassic World: O Mundo dos Dinossauros	BR	None	imdbDisplay	None	0.00
3	tt0369610	13	O Mundo dos Dinossauros	BR	None	None	short title	0.00
4	tt0369610	14	Jurassic World	FR	None	imdbDisplay	None	0.00
...	...	...	...	...	...	...	...	...
331698	tt9827784	2	Sayonara kuchibiru	None	None	original	None	1.00
331699	tt9827784	3	Farewell Song	XWW	en	imdbDisplay	None	0.00
331700	tt9880178	1	La atención	None	None	original	None	1.00
331701	tt9880178	2	La atención	ES	None	None	None	0.00
331702	tt9880178	3	The Attention	XWW	en	imdbDisplay	None	0.00

331703 rows × 8 columns

Using `SQLite3` we will join relevant columns from the table to a dataframe we can use.

In [107]:

```
#reading and joining columns to IMDB dataframe
imdb = pd.read_sql("""
SELECT  movie_basics.movie_id,
        movie_basics.genres,
        movie_basics.runtime_minutes,
        movie_ratings.averagerating,
        movie_akas.title
FROM movie_basics
JOIN movie_ratings ON movie_basics.movie_id = movie_ratings.movie_id
JOIN movie_akas ON movie_ratings.movie_id = movie_akas.movie_id;
""",conn)
imdb
```

Out[107]:

	movie_id	genres	runtime_minutes	averagerating	title
0	tt0063540	Action,Crime,Drama	175.00	7.00	Sangharsh
1	tt0063540	Action,Crime,Drama	175.00	7.00	Sungharsh
2	tt0063540	Action,Crime,Drama	175.00	7.00	Sunghursh
3	tt0063540	Action,Crime,Drama	175.00	7.00	Sunghursh
4	tt0063540	Action,Crime,Drama	175.00	7.00	Sunghursh
...	...	...	...	...	...
261801	tt9905462	Drama	111.00	8.40	Pengalila
261802	tt9905462	Drama	111.00	8.40	Sisterleaf
261803	tt9911774	Drama	130.00	8.40	Padmavyoohathile Abhimanyu
261804	tt9911774	Drama	130.00	8.40	Padmavyuhathile Abhimanyu
261805	tt9911774	Drama	130.00	8.40	Pathmavyuhathile Abhimanyu

261806 rows × 5 columns

Cleaning resulting dataframe

In [108]:

```
#Checking for missing values
print(imdb.info())
print("We find there are missing values as follows:")
print(imdb.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 261806 entries, 0 to 261805
Data columns (total 5 columns):
 #   Column           Non-Null Count   Dtype  
---  ------           --------------   -----  
 0   movie_id         261806 non-null  object 
 1   genres           260621 non-null  object 
 2   runtime_minutes  250553 non-null  float64
 3   averagerating    261806 non-null  float64
 4   title            261806 non-null  object 
dtypes: float64(2), object(3)
memory usage: 10.0+ MB
None
We find there are missing values as follows:
movie_id               0
genres              1185
runtime_minutes    11253
averagerating          0
title                  0
dtype: int64
```

In [109]:

```
#checking for Duplicates
print("Number of duplicated rows:", imdb.movie_id.duplicated().sum())
print("We observe:", imdb.movie_id.duplicated().sum(), "rows; have been duplicated and we drop them below")

#droping duplicated values
imdb.drop_duplicates(subset='movie_id',inplace=True)

print("Resulting dataframe has:", len(imdb), "rows; from the previous 261806" )
imdb
```

```
Number of duplicated rows: 192229
We observe: 192229 rows; have been duplicated and we drop them below
Resulting dataframe has: 69577 rows; from the previous 261806
```

Out[109]:

	movie_id	genres	runtime_minutes	averagerating	title
0	tt0063540	Action,Crime,Drama	175.00	7.00	Sangharsh
5	tt0066787	Biography,Drama	114.00	7.20	Ashad Ka Ek Din
9	tt0069049	Drama	122.00	6.90	Al otro lado del viento
22	tt0069204	Comedy,Drama	NaN	6.10	Sabse Bada Sukh
25	tt0100275	Comedy,Drama,Fantasy	80.00	6.50	La Telenovela Errante
...	...	...	...	...	...
261792	tt9899860	Drama,Thriller	100.00	8.10	Didan in film jorm ast
261795	tt9899880	Comedy	85.00	5.80	Colombos
261797	tt9903952	Comedy,Horror	87.00	9.20	BADMEN with a good behavior
261800	tt9905462	Drama	111.00	8.40	Pengalila
261803	tt9911774	Drama	130.00	8.40	Padmavyoohathile Abhimanyu

69577 rows × 5 columns

We further observe that the data missing in the `movies_id` column is a dulpicate therefore we remove it from our dataset as below

In [110]:

```
#removing the null in movie_id
imdb.movie_id.dropna(inplace=True)
```

It is standard practice to fill missing values with values already in our dataset, which is what we have done here

In [111]:

```
#filling Null values
imdb.fillna(method='ffill',inplace=True)

#checking changes in the dataframe
imdb
```

Out[111]:

	movie_id	genres	runtime_minutes	averagerating	title
0	tt0063540	Action,Crime,Drama	175.00	7.00	Sangharsh
5	tt0066787	Biography,Drama	114.00	7.20	Ashad Ka Ek Din
9	tt0069049	Drama	122.00	6.90	Al otro lado del viento
22	tt0069204	Comedy,Drama	122.00	6.10	Sabse Bada Sukh
25	tt0100275	Comedy,Drama,Fantasy	80.00	6.50	La Telenovela Errante
...	...	...	...	...	...
261792	tt9899860	Drama,Thriller	100.00	8.10	Didan in film jorm ast
261795	tt9899880	Comedy	85.00	5.80	Colombos
261797	tt9903952	Comedy,Horror	87.00	9.20	BADMEN with a good behavior
261800	tt9905462	Drama	111.00	8.40	Pengalila
261803	tt9911774	Drama	130.00	8.40	Padmavyoohathile Abhimanyu

69577 rows × 5 columns

We will then Join the resulting dataframe to our first joined dataframe - `mov_analysis`. We will name this new dataframe `merged_df`. we perform the join using `pandas.merge` method on the `title` column since it appears in both `imdb` and `mov_analysis` dataframes.

In [112]:

```
#performing the join
merged_df = pd.merge(mov_analysis, imdb, left_on="title", right_on="title")
merged_df
```

Out[112]:

	release_date	title	production_budget	domestic_gross	worldwide_gross	studio	foreign_gross	year	month	
0	May 14, 2010	Robin Hood	210000000.00	105487148.00	322459006.00	Par.	216971858.00	2010	05	1
1	Feb 16, 2018	Black Panther	200000000.00	700059566.00	1348258224.00	Sony	648198658.00	2018	02	11
2	Feb 16, 2018	Black Panther	200000000.00	700059566.00	1348258224.00	Sony	648198658.00	2018	02	11
3	Dec 19, 1997	Titanic	200000000.00	659363944.00	2208208395.00	Fox	1548844451.00	1997	12	20
4	Mar 5, 2010	Alice in Wonderland	200000000.00	334191110.00	1025491110.00	LGF	691300000.00	2010	03	8

...	release_date	title	production_budget	domestic_gross	worldwide_gross	studio	foreign_gross	year	month
553	Sep 23, 2011	Weekend	190000.00	484592.00	1577585.00	Fox	1092993.00	2011	09
554	Jul 7, 2017	A Ghost Story	100000.00	1594798.00	2769782.00	Focus	1174984.00	2017	07
555	Mar 18, 2016	Krisha	30000.00	144822.00	144822.00	Sum.	0.00	2016	03
556	Sep 1, 2015	Exeter	25000.00	0.00	489792.00	Uni.	489792.00	2015	09
557	Jul 6, 2001	Cure	10000.00	94596.00	94596.00	Uni.	0.00	2001	07

558 rows × 14 columns

In [113]:

```
#checking the resulting data set how it looks
merged_df.shape
```

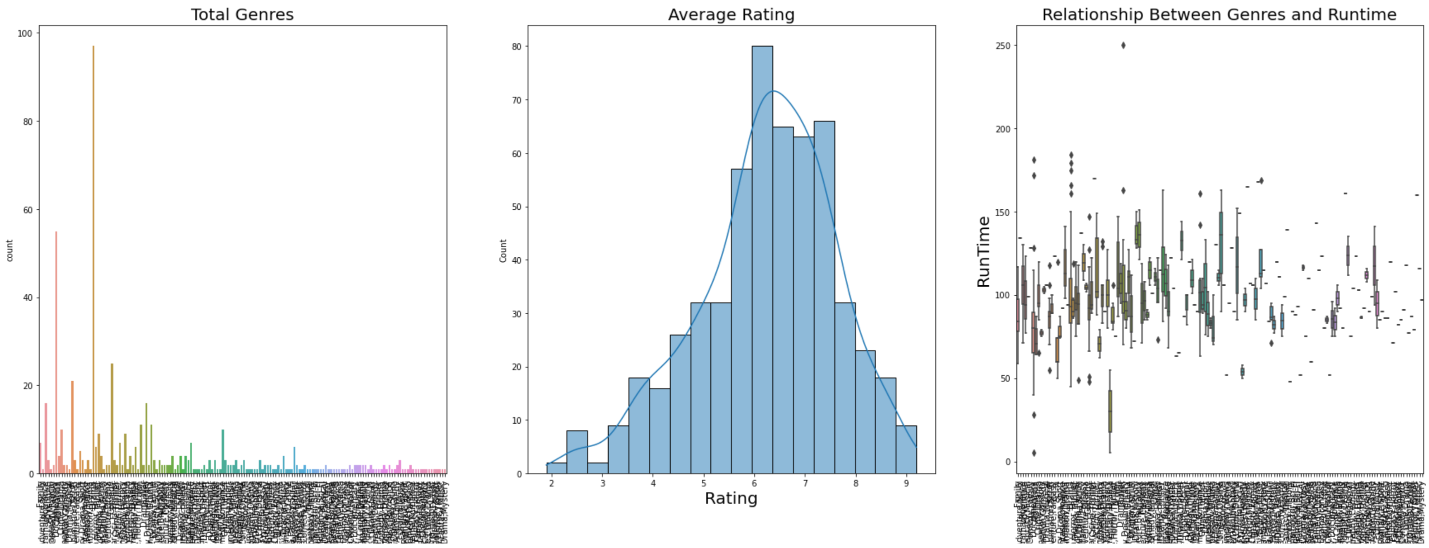
Out[113]:

(558, 14)

Below we will plot several plots that show a summary of our merged dataframe The resulting plots might not be clearly visible, but we only want to see the shape of the plotted data and not neccesarily the content

In [114]:

```
#creating the plot
fig, axes = plt.subplots(ncols=3, figsize=(30,10))

# count plot
sns.countplot(data=merged_df, x='genres', ax=axes[0], )
axes[0].set_title("Total Genres", fontsize=20)
axes[0].tick_params(axis='x', rotation=90)
axes[0].set_xlabel("Genres", fontsize=20)

# histogram
sns.histplot(data=merged_df, x='averagerating', kde=True, ax=axes[1])
axes[1].set_title("Average Rating", fontsize=20)
axes[1].set_xlabel("Rating", fontsize=20)

# box plot
sns.boxplot(data=merged_df, x='genres', y='runtime_minutes', ax=axes[2])
axes[2].set_title("Relationship Between Genres and Runtime", fontsize=20)
axes[2].tick_params(axis='x', rotation=90)
axes[2].set_xlabel("Genres", fontsize=20)
axes[2].set_ylabel("RunTime", fontsize=20)


#displaying the plots
plt.subplots_adjust(hspace=10.5)
plt.show()
```



We might want to group the data using the genres column in order to carry out analysis on the runtimes and ratings for each genre. we wll name the new sub set as `grouped studio`

In [115]:

```
#we will group the data as follows
grouped_studio = merged_df.groupby('studio')
#to have data displayed ina dataframe we would need to aggregate the values in the other columns.
studio_data = grouped_studio.mean()[['runtime_minutes', 'averagerating']]
studio_data = studio_data.reset_index(0)
studio_data
```

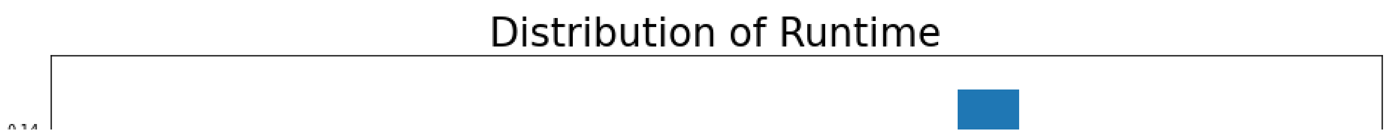
Out[115]:

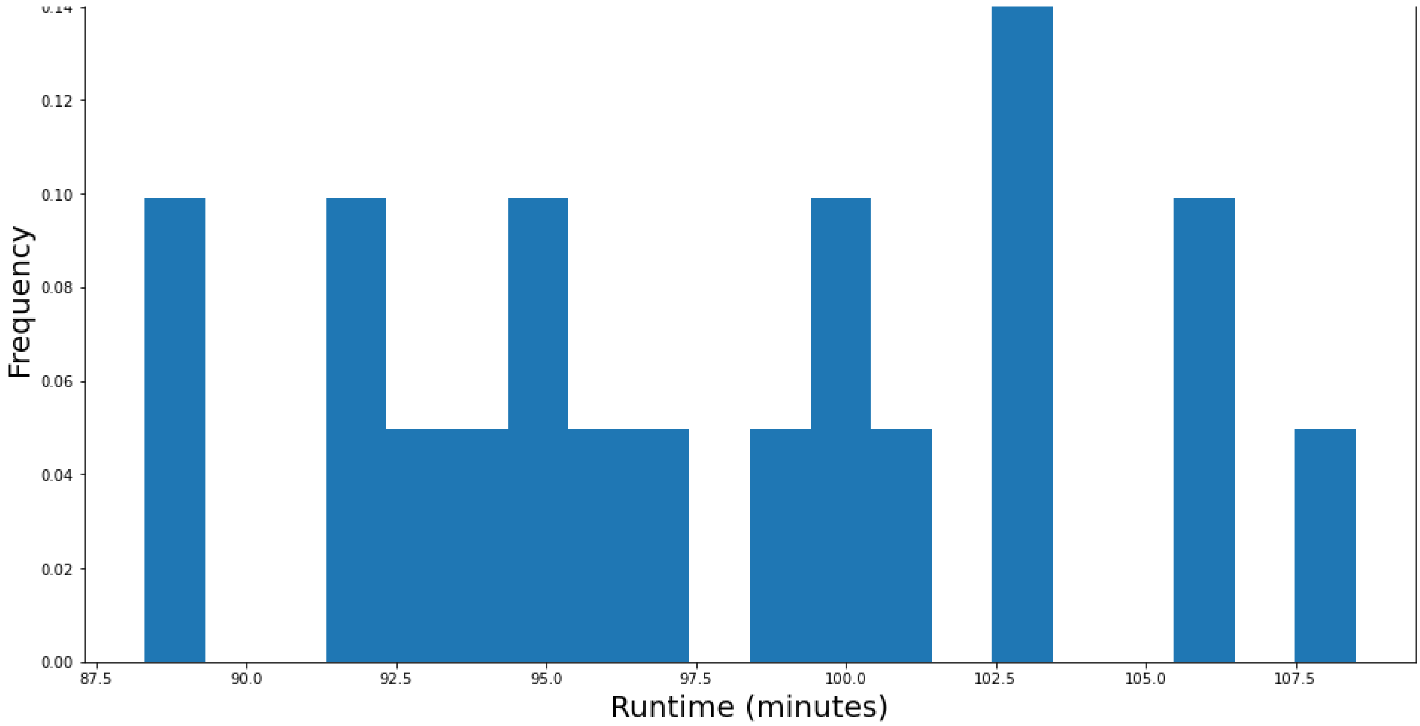
	studio	runtime_minutes	averagerating
0	BV	105.74	6.32
1	CBS	103.00	4.00
2	CL	95.77	6.48
3	Focus	108.50	7.25
4	Fox	96.97	6.32
5	FoxS	102.79	6.78
6	LGF	94.55	6.41
7	MBox	99.73	6.33
8	MGM	91.89	6.70
9	Over.	91.71	5.43
10	P/DW	88.30	5.61
11	Par.	94.37	6.39
12	SGem	100.10	6.22
13	Sony	92.42	5.91
14	Sum.	88.78	6.03
15	Uni.	100.47	6.07
16	W/Dim.	94.00	6.64
17	WB	103.03	6.24
18	WB (NL)	98.42	6.38
19	Wein.	105.60	6.00

This is a plot that shows the `average runtime` for movies.

In [116]:

```
# Create a histogram of runtime
plt.figure(figsize=(16,9))
plt.hist(studio_data['runtime_minutes'], density=True, bins=20)
plt.xlabel('Runtime (minutes)',fontsize=20)
plt.ylabel('Frequency',fontsize=20)
plt.title('Distribution of Runtime',fontsize=26)
plt.show()
```





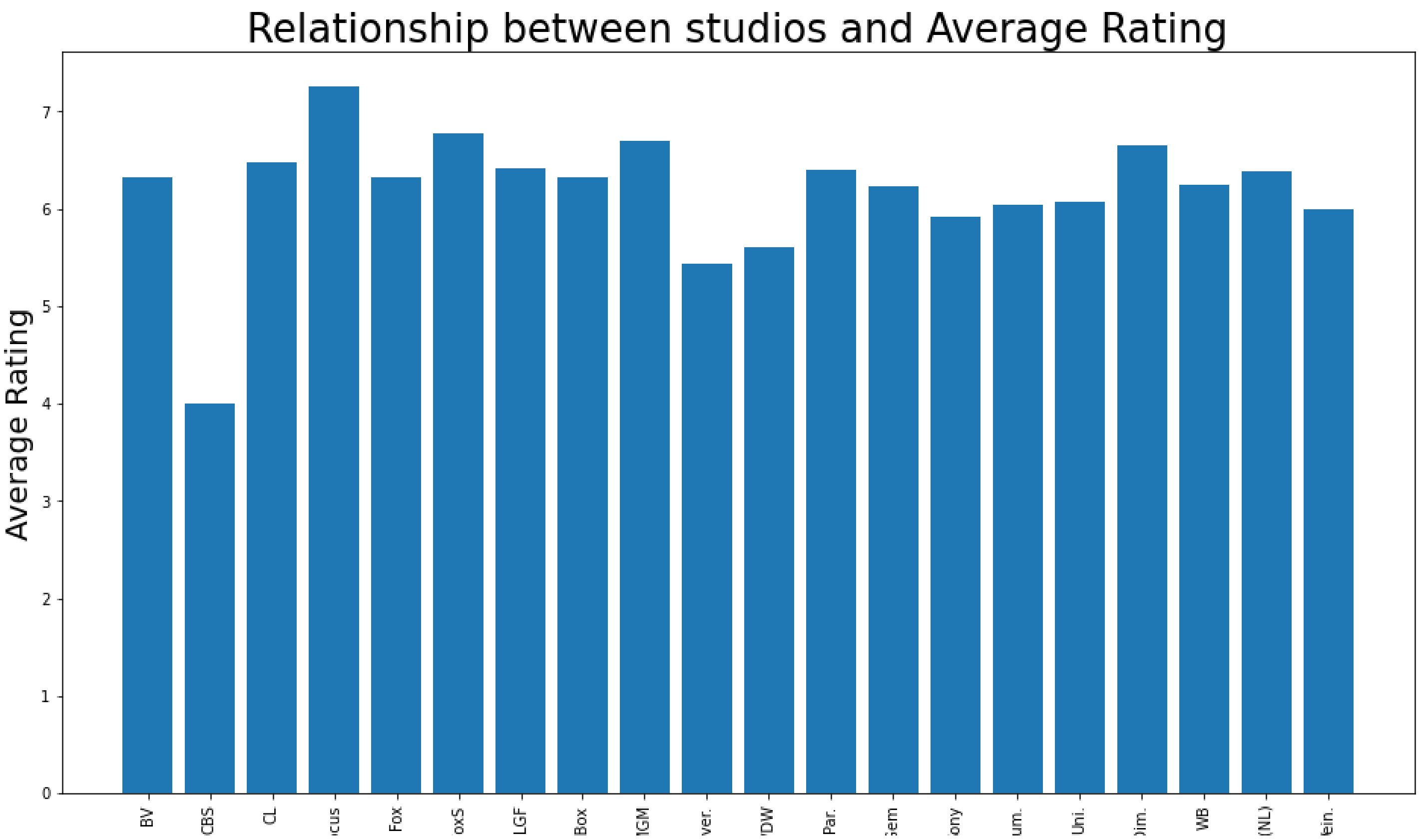
2. How does the rating of a movie affect its revenue at the box office?

To address this question, we can create a `histogram plot` **showing the** `relationship between studio and average rating for movies`**.**

This is a plot that shows the relationship between `studio` **and** `average rating` **for movies.**

In [117]:

```
# Create a bar plot of studio in relation to the average rating
plt.figure(figsize=(16,9))
plt.bar(studio_data['studio'], height=studio_data['averagerating'])
plt.xlabel('Studio',fontsize=20)
plt.ylabel('Average Rating',fontsize=20)
plt.title('Relationship between studios and Average Rating',fontsize=26)
plt.xticks(rotation=90)
plt.show()
```



We observe that the highest rated studio is Focus studio, while the Highest earning studio was WB even the second rated Fox studios did not also garner good ratings. We would need to see personl movie reviews to understand why this change is presented in our data.

In [118]:

```
#grouping genre and average rating
#we will group the data as follows
grouped1_data = merged_df.groupby('genres')
#to have data displayed ina dataframe we would need to aggregate the values in the other columns.
genre_data = grouped1_data.sum()[['runtime_minutes','averagerating']].sort_values(by='averagerating',ascending=False)
genre_data = genre_data.reset_index(0)
genre_data
```

Out[118]:

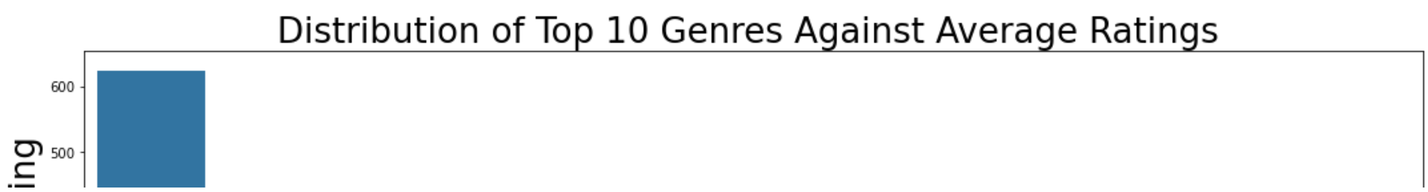
	genres	runtime_minutes	averagerating
0	Drama	9694.00	622.80
1	Documentary	4380.00	388.80
2	Comedy	2311.00	152.30
3	Horror	1806.00	103.60
4	Drama,Romance	1684.00	97.90
...	...	...	...
149	Action,Drama,Romance	143.00	3.60
150	Comedy,Romance,Sport	90.00	2.90
151	Comedy,Mystery,Sci-Fi	88.00	2.70
152	Comedy,Family	106.00	2.50
153	Action,Comedy,Horror	91.00	2.40

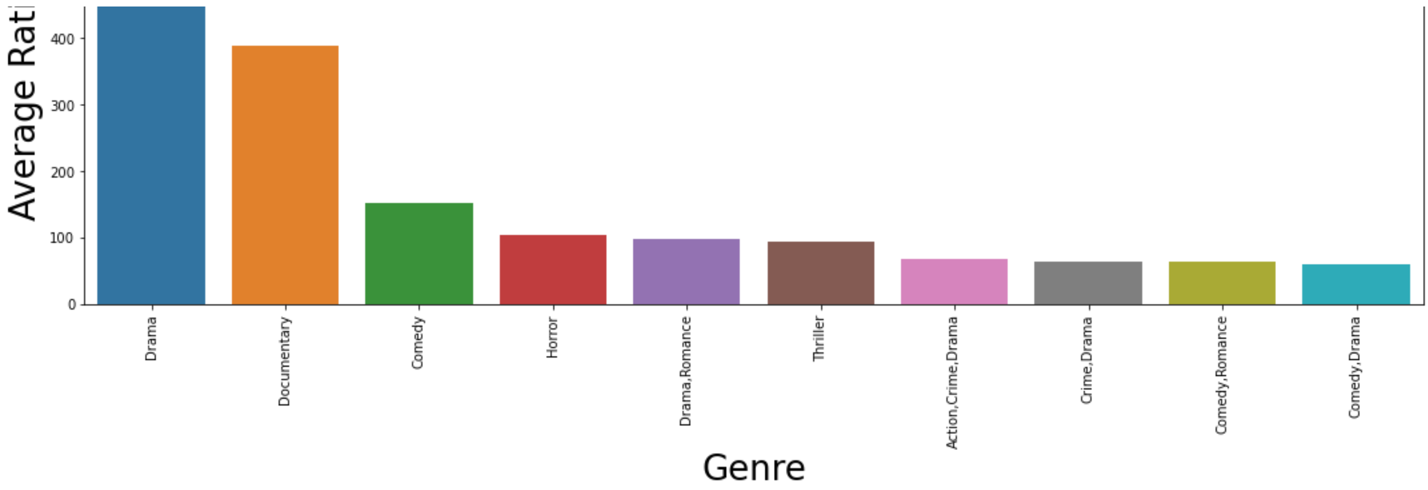
154 rows × 3 columns

We can also draw a bar graph showing the distribution of average ratings against genre Using the combined data,we observe a very unclear plot is presented. This can easily be corrected by creating a subset from containing only genre and rating this still does not get us the desired results, we therefore sort the data in descending order, and only plot the highest rated movies

In [119]:

```
# Create a box plot of average ratings by genre

y_to_plot = genre_data['averagerating'].head(10)
x_to_plot = genre_data['genres'].head(10)

plt.figure(figsize=(18,6))
sns.barplot(x=x_to_plot, y=y_to_plot,  data=genre_data)
plt.xticks(rotation=90)
plt.xlabel('Genre',fontsize=26)
plt.ylabel('Average Rating',fontsize=26)
plt.title('Distribution of Top 10 Genres Against Average Ratings ',fontsize=26)
plt.show()
```





We observe the highest earninggenre is `Drama` , followed closely by `Documentary` , `comedy` , `Horror` , `Drama/romance` , `thriller` . But we also observe that some columns contain elements of another genre, but wewere unable to create a function to separate the same.

Here we will plot the relationship between `Genres` and `Revenue` and `profit` versus `cost of production` . The data will be subset as follows, named `grouped Genres`

In [120]:

```
#grouping genre and average rating
#we will group the data as follows
grouped_genres = merged_df.groupby('genres')
#to have data displayed ina dataframe we would need to aggregate the values in the other columns.
genre_earn = grouped_genres.mean()[['production_budget','worldwide_gross','profit']]#.sort_values(by='worldwide_gross' ascending=False, inplace=True)
genre_earn = genre_earn.reset_index(0)
genre_earn
```

Out[120]:

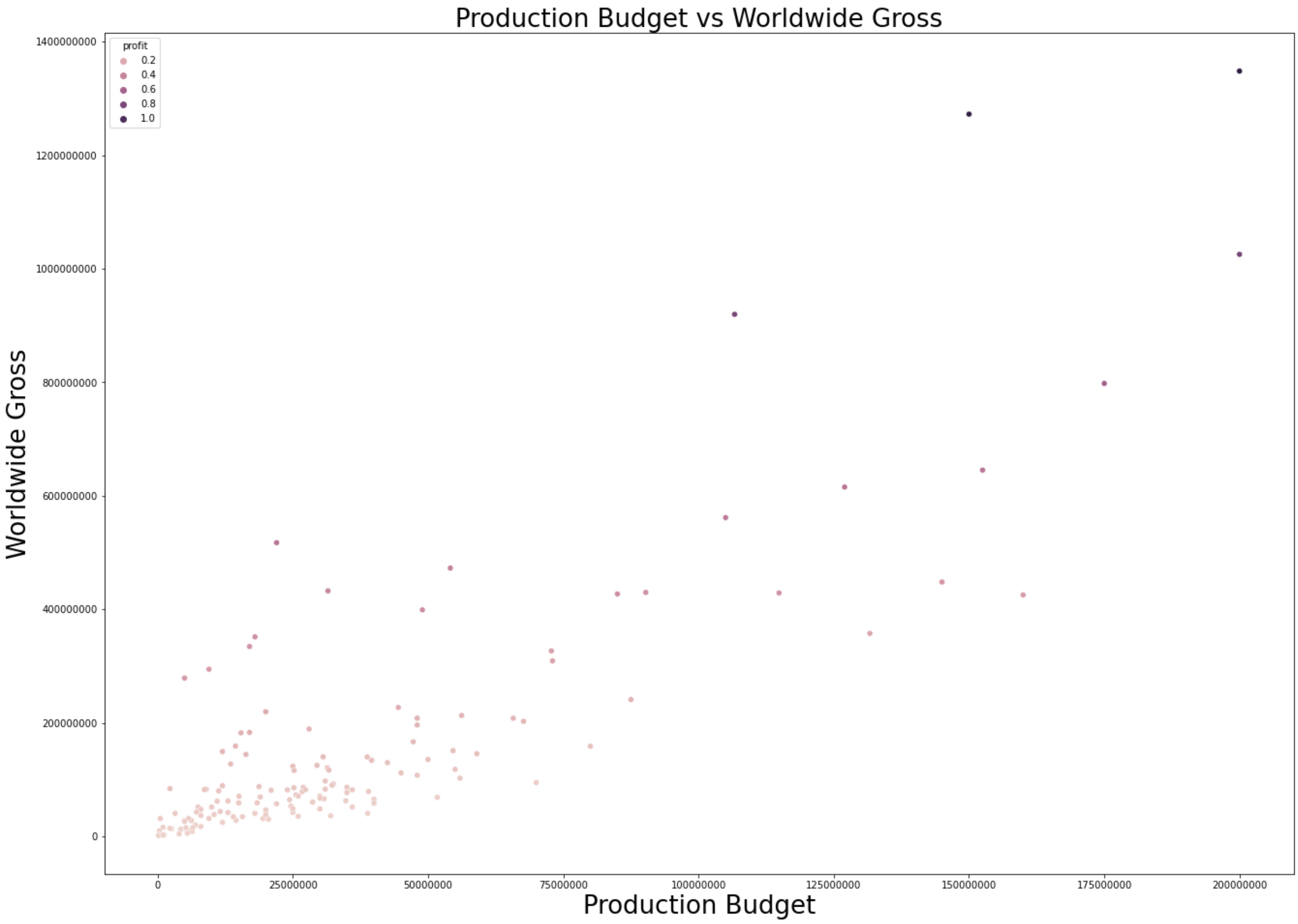
	genres	production_budget	worldwide_gross	profit
0	Action	31409090.91	120767464.45	89358373.55
1	Action,Adventure,Biography	42500000.00	129941054.00	87441054.00
2	Action,Adventure,Comedy	31500000.00	432633203.00	401133203.00
3	Action,Adventure,Drama	87500000.00	241156904.00	153656904.00
4	Action,Adventure,Fantasy	160000000.00	425522281.00	265522281.00
...	...	...	...	...
149	Musical	20000000.00	38164784.00	18164784.00
150	Mystery	36000000.00	51876453.00	15876453.00
151	Romance	6500000.00	15428789.00	8928789.00
152	Sci-Fi	72800000.00	326914580.20	254114580.20
153	Thriller	26017978.12	71123767.12	45105789.00

154 rows × 4 columns

In [121]:

```
#We will begin with a Scatter plot to show the relationship between production_budget and worldwide_gross

plt.figure(figsize=(22,16))
sns.scatterplot(data=genre_earn, x='production_budget', y='worldwide_gross', hue='profit')
plt.title('Production Budget vs Worldwide Gross',fontsize=26)
```

```
plt.xlabel('Production Budget',fontsize=26)
plt.ylabel('Worldwide Gross',fontsize=26)
plt.ticklabel_format(style='plain',axis='both')
plt.show()
```



Subsets for Key Performance Indicators

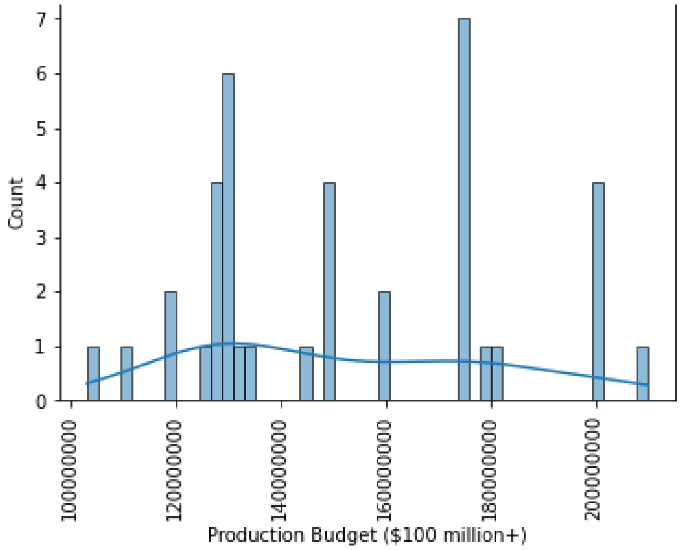
Some of the Key performance indicators include:

1. **High-budget films: films with a production budget above a certain threshold (e.g. $100 million)**
2. **Successful films: films with a high domestic and/or worldwide gross (e.g. in the top quartile)**
3. **Studio-specific subsets: subsets for each major studio (e.g. Disney, Warner Bros., Universal, etc.)**
4. **Genre-specific subsets: subsets for each major film genre (e.g. action, comedy, drama, etc.)**
5. **Time-specific subsets: subsets for specific time periods (e.g. decade, year)**
6. **International subsets: subsets focused on non-U.S. markets, such as films that performed well in specific countries or regions.**

In [122]:

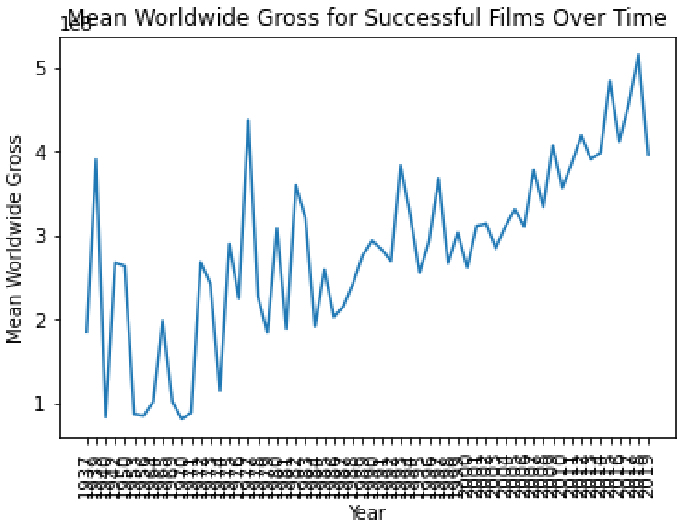
```
#first KPI
#High-budget films
high_budget_films = merged_df[merged_df['production_budget'] > 100000000]
high_budget_films

# create histogram of production budget distribution
sns.histplot(data=high_budget_films, x='production_budget', bins=50, kde=True)
plt.title('Distribution of Production Budget for High-Budget Films')
plt.xlabel('Production Budget ($100 million+)')
plt.ylabel('Count')
plt.ticklabel_format(style='plain', axis='x')
plt.xticks(rotation=90)
plt.show()
```

Distribution of Production Budget for High-Budget Films



In [123]:

```
#Second KPI
#Successful films
domestic_quartile = mov_analysis[mov_analysis['domestic_gross'] >= mov_analysis['domesti
c_gross'].quantile(0.75)]
worldwide_quartile = mov_analysis[mov_analysis['worldwide_gross'] >= mov_analysis['world
wide_gross'].quantile(0.75)]
successful_films = pd.concat([domestic_quartile, worldwide_quartile]).drop_duplicates()

#Ploting
successful_films_mean_gross = successful_films.groupby('year')['worldwide_gross'].mean()
.reset_index()
sns.lineplot(data=successful_films_mean_gross, x='year', y='worldwide_gross')
plt.title('Mean Worldwide Gross for Successful Films Over Time')
plt.xlabel('Year')
plt.ylabel('Mean Worldwide Gross')
plt.xticks(rotation=90)
plt.show()
```



In [124]:

```
#Third KPI
#Studio-specific subsets
disney = mov_analysis[mov_analysis['studio'] == 'Disney']
warner_bros = mov_analysis[mov_analysis['studio'] == 'Warner Bros.']
universal = mov_analysis[mov_analysis['studio'] == 'Universal']
paramount = mov_analysis[mov_analysis['studio'] == 'Paramount Pictures']
sony = mov_analysis[mov_analysis['studio'] == 'Sony Pictures']
```

In [125]:

```
#Fourth KPI
#Genre-specific subsets:
```

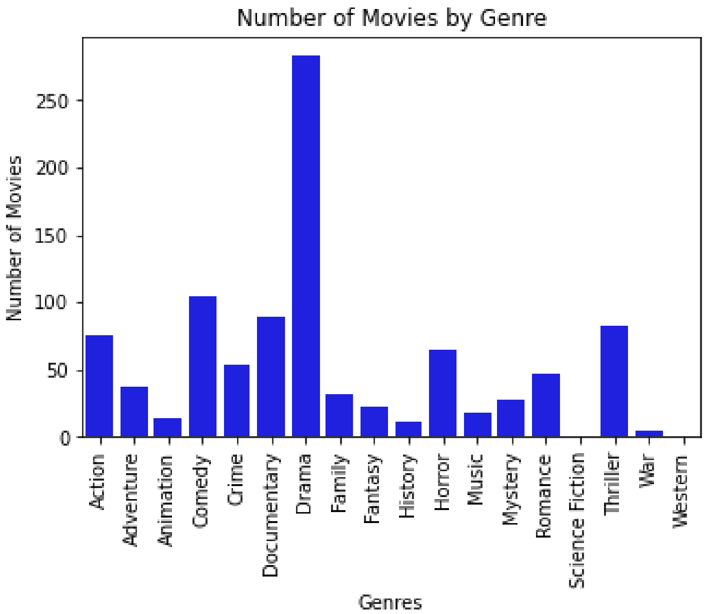
```
# Action
action_movies = merged_df[merged_df['genres'].str.contains('Action')]

# Adventure
adventure_movies = merged_df[merged_df['genres'].str.contains('Adventure')]

# Animation
animation_movies = merged_df[merged_df['genres'].str.contains('Animation')]

# Comedy
comedy_movies = merged_df[merged_df['genres'].str.contains('Comedy')]

# Crime
crime_movies = merged_df[merged_df['genres'].str.contains('Crime')]

# Documentary
documentary_movies = merged_df[merged_df['genres'].str.contains('Documentary')]

# Drama
drama_movies = merged_df[merged_df['genres'].str.contains('Drama')]

# Family
family_movies = merged_df[merged_df['genres'].str.contains('Family')]

# Fantasy
fantasy_movies = merged_df[merged_df['genres'].str.contains('Fantasy')]

# History
history_movies = merged_df[merged_df['genres'].str.contains('History')]

# Horror
horror_movies = merged_df[merged_df['genres'].str.contains('Horror')]

# Music
music_movies = merged_df[merged_df['genres'].str.contains('Music')]

# Mystery
mystery_movies = merged_df[merged_df['genres'].str.contains('Mystery')]

# Romance
romance_movies = merged_df[merged_df['genres'].str.contains('Romance')]

# Science Fiction
sci_fi_movies = merged_df[merged_df['genres'].str.contains('Science Fiction')]

# Thriller
thriller_movies = merged_df[merged_df['genres'].str.contains('Thriller')]

# War
war_movies = merged_df[merged_df['genres'].str.contains('War')]

# Western
western_movies = merged_df[merged_df['genres'].str.contains('Western')]
```

Here we will create a bar chart using the above subset data in order to get the number of movies in each genre. This will help assertain the findings from the plot with ratings and genre

In [126]:

```
#first, we Create a list of all genres
genres = ['Action', 'Adventure', 'Animation', 'Comedy', 'Crime',
          'Documentary', 'Drama', 'Family', 'Fantasy', 'History',
          'Horror', 'Music', 'Mystery', 'Romance', 'Science Fiction',
          'Thriller', 'War', 'Western']

# Next, a list of counts for each genre
counts = [len(action_movies), len(adventure_movies), len(animation_movies),
          len(comedy_movies), len(crime_movies), len(documentary_movies),
          len(drama_movies), len(family_movies), len(fantasy_movies),
          len(history_movies), len(horror_movies), len(music_movies),
```

```
          len(mystery_movies), len(romance_movies), len(sci_fi_movies),
          len(thriller_movies), len(war_movies), len(western_movies)]

# Create the bar plot using Seaborn
ax = sns.barplot(x=genres, y=counts, color='b')

# Customize the plot
plt.xlabel('Genres')
plt.ylabel('Number of Movies')
plt.title('Number of Movies by Genre')
plt.xticks(rotation=90)


# Show the plot
plt.show()
```



Conclusion

We have observed form all the above illustrations that indeed, the movie production business is very lucrative. For microsoft to consider investing in their own studio, it will be a significant diversification of their revenue base as long as they are willing to invest the funds required. This is just a preliminary analysis and a much deeper understandind will be required to provide a detailed course of action to take. This is because we will have to take intoconsideration the whole process of movie creation from selecting the best directors, identifying script writers, looking into the relationship between actors and movie ratings. we would also need to consider viewer feedback on each movie released and its relationship to revenue and by extension production budget. Looking forward to be contracted to carry out this further analysis