Project Overview. ¶

For this project, I will use exploratory data analysis to generate insights for a business stakeholder.

Business Problem.

Microsoft sees all the big companies creating original video content and they want to get in on the fun. They have decided to create a new movie studio, but they don't know anything about creating movies. You are charged with exploring what types of films are currently doing the best at the box office. You must then translate those findings into actionable insights that the head of Microsoft's new movie studio can use to help decide what type of films to create.

Research Objectives.

1.Identify the types of films that are currently doing the best at the box office.

2.Analyze the characteristics of these successful films, including their genre, budget, release date, target audience, and critical reception.

Methodology.

To achieve our research objectives, we will use exploratory data analysis to analyze data on films that have been released in recent years, including their box office performance, genre, budget, release date, target audience, and critical reception. We will use statistical analysis and data visualization techniques to identify patterns and trends in the data.

Expected Outcomes.

Based on our research, we expect to identify the types of films that are currently successful at the box office, as well as the characteristics that contribute to their success. We will provide actionable insights for Microsoft's new movie studio to use in creating films that are more likely to resonate with audiences and generate revenue.

Recording the Experimental Design.

.upload and read our csv files

.clean our dataset

.perfom EDA

.Conclusion

.Recommendation

Data Loading and Preprocessing.

To start with the analysis, we first need to import the necessary libraries and load the dataset into our Jupyter Notebook. Here's how you can do that:

Importing Libraries.

We will be using the following libraries for this project:

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
```

Load the Datasets.

To load the dataset into a pandas dataframe, follow these steps:

1. Make sure that you have downloaded the dataset in gzip format.
2. Use the `read_csv` function to load the dataset into a pandas dataframe. Since the dataset is compressed in gzip format, we need to use the `compression` parameter of the `read_csv` function to specify the compression method. Here's how you can do it:

In [3]:
```
bom_df = pd.read_csv('bom.movie_gross.csv.gz', compression='gzip') #load th
tn_df = pd.read_csv('tn.movie_budgets.csv.gz', compression='gzip') #load th
tmdb_df = pd.read_csv('tmdb.movies.csv.gz', compression='gzip') #load the m
```

Previewing the top 5 rows of each dataset.

In [4]:
```
print(bom_df.head()) #displays the first 5 rows of the dataframe to check t
```

```
                                     title studio  domestic_gross  \
0                                Toy Story 3     BV     415000000.0
1                  Alice in Wonderland (2010)     BV     334200000.0
2  Harry Potter and the Deathly Hallows Part 1     WB     296000000.0
3                                  Inception     WB     292600000.0
4                        Shrek Forever After   P/DW     238700000.0

  foreign_gross  year
0     652000000  2010
1     691300000  2010
2     664300000  2010
3     535700000  2010
4     513900000  2010
```

Observation.

The "bom_df" (Box Office Mojo dataset) contains information on movie titles, studios, domestic and foreign gross earnings, and the year of release.

In [5]:
```
print(tn_df.head()) #displays the first 5 rows of the dataframe to check th
```

```
   id  release_date                                        movie  \
0   1  Dec 18, 2009                                       Avatar
1   2  May 20, 2011  Pirates of the Caribbean: On Stranger Tides
2   3   Jun 7, 2019                                  Dark Phoenix
3   4   May 1, 2015                  Avengers: Age of Ultron
4   5  Dec 15, 2017           Star Wars Ep. VIII: The Last Jedi

  production_budget domestic_gross worldwide_gross
0      $425,000,000   $760,507,625  $2,776,345,279
1      $410,600,000   $241,063,875  $1,045,663,875
2      $350,000,000    $42,762,350    $149,762,350
3      $330,600,000   $459,005,868  $1,403,013,963
4      $317,000,000   $620,181,382  $1,316,721,747
```

Observation.

The "tn_df" (The Numbers dataset) above contains information on movie titles, release dates, production budgets, domestic and worldwide gross earnings.

In [6]:
```
print(tmdb_df.head()) #displays the first 5 rows of the dataframe to check
```

```
   Unnamed: 0            genre_ids     id original_language  \
0           0      [12, 14, 10751]  12444                en
1           1  [14, 12, 16, 10751]  10191                en
2           2        [12, 28, 878]  10138                en
3           3      [16, 35, 10751]    862                en
4           4        [28, 878, 12]  27205                en

                             original_title  popularity release_date
\
0  Harry Potter and the Deathly Hallows: Part 1      33.533   2010-11-19
1                    How to Train Your Dragon      28.734   2010-03-26
2                                  Iron Man 2      28.515   2010-05-07
3                                   Toy Story      28.005   1995-11-22
4                                   Inception      27.920   2010-07-16

                                       title  vote_average  vote_count
0  Harry Potter and the Deathly Hallows: Part 1           7.7       10788
1                    How to Train Your Dragon           7.7        7610
2                                  Iron Man 2           6.8       12368
3                                   Toy Story           7.9       10174
4                                   Inception           8.3       22186
```

Observation.

The "tmdb_df" (The Movie Database dataset) above contains information on movie genres, original language, popularity, release date, titles, and voting scores.

We can also see that the data in each dataset is organized differently, with different column names and formats.

Checking the data.

In [7]:
```
print("The box office mojo shape",bom_df.shape)
print("The numbers shape",tn_df.shape)
print("The movie database shape",tmdb_df.shape)
```

```
The box office mojo shape (3387, 5)
The numbers shape (5782, 6)
The movie database shape (26517, 10)
```

Observation.

The bom_df dataframe has 3387 rows and 5 columns. The tn_df dataframe has 5782 rows and 6 columns. The tmdb_df dataframe has 26517 rows and 10 columns.

These outputs show the dimensions of each of the dataframes, which gives us an idea of the amount of data we are dealing with in each case.

Previewing the bottom of our datasets.

In [8]:
```
# Previewing the bottom of bom_df
bom_df.tail()
```

Out[8]:

	title	studio	domestic_gross	foreign_gross	year
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

In [9]:
```
# Previewing the bottom of tn_df
tn_df.tail()
```

Out[9]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
5777	78	Dec 31, 2018	Red 11	$7,000	$0	$0
5778	79	Apr 2, 1999	Following	$6,000	$48,482	$240,495
5779	80	Jul 13, 2005	Return to the Land of Wonders	$5,000	$1,338	$1,338
5780	81	Sep 29, 2015	A Plague So Pleasant	$1,400	$0	$0
5781	82	Aug 5, 2005	My Date With Drew	$1,100	$181,041	$181,041

In [10]:
```
# Previewing the bottom of tmdb_df
tmdb_df.tail()
```

Out[10]:

	Unnamed: 0	genre_ids	id	original_language	original_title	popularity	releas
26512	26512	[27, 18]	488143	en	Laboratory Conditions	0.6	2018
26513	26513	[18, 53]	485975	en	_EXHIBIT_84xxx_	0.6	2018
26514	26514	[14, 28, 12]	381231	en	The Last One	0.6	2018
26515	26515	[10751, 12, 28]	366854	en	Trailer Made	0.6	2018
26516	26516	[53, 27]	309885	en	The Church	0.6	2018

Observations.

From the output of bom_df.tail(), we can observe that the dataset contains information about movie titles, their respective studios, domestic gross, foreign gross, and year of release. We can also see that some of the rows have missing values for foreign gross.

From the output of tn_df.tail(), we can observe that the dataset contains information about movie IDs, release dates, movie titles, production budgets, domestic gross, and worldwide gross.we can see that the "worldwide_gross" column has values of 0 for some movies, even though they have a non-zero value for both "production_budget" and "domestic_gross". This seems like a data entry error and may need to be corrected.

From the output of tmdb_df.tail(), we can observe that the dataset contains information about movie titles, their original titles, genre IDs, release dates, popularity scores, vote averages, vote counts, and original language. We can also see that the dataset contains a column named Unnamed: 0 which seems to be an extra index column.

We shall fix/clean the above errors and inconsistencies in the following section.

Checking whether each column has an appropriate datatype.

In [11]:
```
# Checking datatypes of columns in bom_df
bom_df.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
```

In [12]:
```
# Checking datatypes of columns in tn_df
tn_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5782 entries, 0 to 5781
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   id                 5782 non-null   int64 
 1   release_date       5782 non-null   object
 2   movie              5782 non-null   object
 3   production_budget  5782 non-null   object
 4   domestic_gross     5782 non-null   object
 5   worldwide_gross    5782 non-null   object
dtypes: int64(1), object(5)
memory usage: 271.2+ KB
```

In [13]:
```
# Checking datatypes of columns in tmdb_df
tmdb_df.info()
```

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

Observations.

The info() method is displaying the correct number of non-null values and datatypes for each column in the three datasets.

Checking for duplicate rows.

In [14]:
```
# Checking for duplicates in bom_df
print("Number of duplicate rows in bom_df:", bom_df.duplicated().sum())
print("-"*30)
# Checking for duplicates in tn_df
print("Number of duplicate rows in tn_df:", tn_df.duplicated().sum())
print("-"*30)
# Checking for duplicates in tmdb_df
print("Number of duplicate rows in tmdb_df:", tmdb_df.duplicated().sum())
```

```
Number of duplicate rows in bom_df: 0
------------------------------
Number of duplicate rows in tn_df: 0
------------------------------
Number of duplicate rows in tmdb_df: 0
```

Observations.

No duplicate rows in all 3 datasets.

Checking for missing values.

In [15]:
```
# Checking for missing values in bom_df
print("Number of missing values in bom_df:\n", bom_df.isnull().sum())
print("-"*30)
# Checking for missing values in tn_df
print("Number of missing values in tn_df:\n", tn_df.isnull().sum())
print("-"*30)
# Checking for missing values in tmdb_df
print("Number of missing values in tmdb_df:\n", tmdb_df.isnull().sum())
```

```
Number of missing values in bom_df:
 title               0
studio               5
domestic_gross      28
foreign_gross     1350
year                 0
dtype: int64
------------------------------
Number of missing values in tn_df:
 id                   0
release_date         0
movie                0
production_budget    0
domestic_gross       0
worldwide_gross      0
dtype: int64
------------------------------
Number of missing values in tmdb_df:
 Unnamed: 0           0
genre_ids            0
id                   0
original_language    0
original_title       0
popularity           0
release_date         0
title                0
vote_average         0
vote_count           0
dtype: int64
```

Observations.

The bom_df dataset has missing values in the studio, domestic_gross, and foreign_gross columns.

We shall fix this in the next section.

Checking for inconsistent values.

In [16]:
```
# Checking for inconsistent values in bom_df
print("Number of inconsistent values in bom_df:\n", bom_df.select_dtypes(ir
print("-"*30)
# Checking for inconsistent values in tn_df
print("Number of inconsistent values in tn_df:\n", tn_df.select_dtypes(incl
print("-"*30)
# Checking for inconsistent values in tmdb_df
print("Number of inconsistent values in tmdb_df:\n", tmdb_df.select_dtypes(
```

```
Number of inconsistent values in bom_df:
 title            3386
studio             257
foreign_gross     1204
dtype: int64
------------------------------
Number of inconsistent values in tn_df:
 release_date         2418
movie                5698
production_budget     509
domestic_gross       5164
worldwide_gross      5356
dtype: int64
------------------------------
Number of inconsistent values in tmdb_df:
 genre_ids             2477
original_language       76
original_title       24835
release_date          3433
title                24688
dtype: int64
```

Observations.

The output above shows the number of unique values in each object column of the three dataframes. This can be used to identify inconsistent values in the data. For example, in bom_df, there are 257 unique values for the "studio" column, which suggests that there may be inconsistent naming or spelling of studios. Similarly, in tn_df, there are 5698 unique values for the "movie" column, which could indicate inconsistent naming or spelling of movie titles. In tmdb_df, there are 2477 unique values for the "genre_ids" column, which may indicate inconsistent or incorrect genre tagging for movies.

We shall fix this in the next section.

Checking for outliers.

In [17]:
```
# Checking for outliers in bom_df
print("Statistics of domestic_gross column in bom_df:\n", bom_df["domestic_
print("-"*30)
# Checking for outliers in tn_df
print("Statistics of production_budget column in tn_df:\n", tn_df["producti
print("-"*30)
# Checking for outliers in tmdb_df
print("Statistics of popularity column in tmdb_df:\n", tmdb_df["popularity"
```

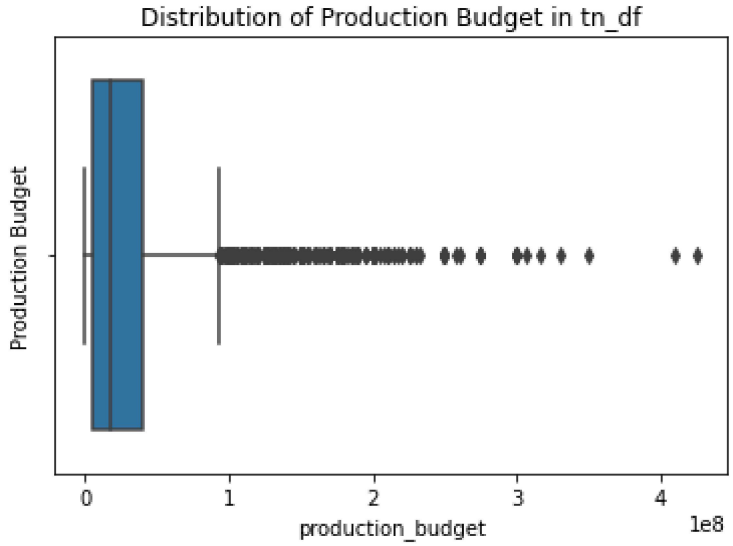
```
Statistics of domestic_gross column in bom_df:
 count    3.359000e+03
mean     2.874585e+07
std      6.698250e+07
min      1.000000e+02
25%      1.200000e+05
50%      1.400000e+06
75%      2.790000e+07
max      9.367000e+08
Name: domestic_gross, dtype: float64
------------------------------
Statistics of production_budget column in tn_df:
 count            5782
unique            509
top       $20,000,000
freq              231
Name: production_budget, dtype: object
------------------------------
Statistics of popularity column in tmdb_df:
 count    26517.000000
mean         3.130912
std          4.355229
min          0.600000
25%          0.600000
50%          1.374000
75%          3.694000
max         80.773000
Name: popularity, dtype: float64
```

Observations.

In bom_df, the "domestic_gross" column has a large range of values, with the minimum value of 1 and the maximum value of 936,700,000. This suggests the presence of outliers in this column. In tn_df, the "production_budget" column has a minimum value of 0 and is also represented as an object instead of a numeric data type, which could indicate the presence of inconsistent or incorrect data. The fact that there are only 509 unique values for this column out of 5782 total values could also indicate some level of inconsistency in the data. In tmdb_df, the "popularity" column has a large range of values, with the maximum value of 80.773. This suggests the presence of outliers in this column.

In [18]:
```
tn_df["production_budget"] = pd.to_numeric(tn_df["production_budget"].astyp
# We can convert the production_budget column in tn_df to a numeric data ty
#function with this line of code
```

In [19]:
```
import seaborn as sns

# Plot boxplot of production_budget column in tn_df
sns.boxplot(x=tn_df["production_budget"], orient="horizontal")
plt.title("Distribution of Production Budget in tn_df")
plt.ylabel("Production Budget")
plt.show()
```



Observations.

The boxplot of production_budget in tn_df shows that there are a lot of outliers above the upper whisker, indicating that there are some movies with very high production budgets. The median is around $20 million, and the interquartile range (IQR) is between 5 million and $45 million, indicating that most movies have a production budget within this range.

However, I am more interested in the general trends and patterns in the data, the outliers may not be as important, and I choose to leave them as they are.

Data Cleaning.

1. Cleaning the box office mojo dataset.

In [20]:
```
# drop rows with missing values in relevant columns
bom_df = bom_df.dropna(subset=['domestic_gross', 'studio'])

# fill missing values in foreign_gross column with 0
bom_df['foreign_gross'] = bom_df['foreign_gross'].fillna(0)

# convert the domestic_gross and foreign_gross columns to numeric
bom_df[['domestic_gross', 'foreign_gross']] = bom_df[['domestic_gross', 'fo

# create a new column for worldwide gross by adding domestic and foreign gr
bom_df['worldwide_gross'] = bom_df['domestic_gross'] + bom_df['foreign_gros

# drop irrelevant columns
bom_df = bom_df.drop(columns=['domestic_gross', 'foreign_gross'])

# preview the first 5 rows of the cleaned dataset
bom_df.head()
```

Out[20]:

	title	studio	year	worldwide_gross
0	Toy Story 3	BV	2010	1.067000e+09
1	Alice in Wonderland (2010)	BV	2010	1.025500e+09
2	Harry Potter and the Deathly Hallows Part 1	WB	2010	9.603000e+08
3	Inception	WB	2010	8.283000e+08
4	Shrek Forever After	P/DW	2010	7.526000e+08

2.Cleaning the numbers dataset.

In [21]:
```
# Drop rows with 0 value in "worldwide_gross"
tn_df = tn_df[tn_df['worldwide_gross'] != 0]
```

In [22]:
```
# preview the first few rows of the modified dataset
print(tn_df.head())
print("-"*30)
# preview the last few rows of the modified dataset
print(tn_df.tail())
print("-"*30)
# check the summary statistics of the dataset
print(tn_df.describe())
```

```
   id  release_date                                        movie  \
0   1  Dec 18, 2009                                       Avatar
1   2  May 20, 2011  Pirates of the Caribbean: On Stranger Tides
2   3   Jun 7, 2019                                  Dark Phoenix
3   4   May 1, 2015                  Avengers: Age of Ultron
4   5  Dec 15, 2017           Star Wars Ep. VIII: The Last Jedi

   production_budget domestic_gross worldwide_gross
0          425000000   $760,507,625  $2,776,345,279
1          410600000   $241,063,875  $1,045,663,875
2          350000000    $42,762,350    $149,762,350
3          330600000   $459,005,868  $1,403,013,963
4          317000000   $620,181,382  $1,316,721,747
------------------------------
      id  release_date                          movie  production_budget
\
5777  78  Dec 31, 2018                       Red 11                7000
5778  79   Apr 2, 1999                    Following                6000
5779  80  Jul 13, 2005  Return to the Land of Wonders                5000
5780  81  Sep 29, 2015           A Plague So Pleasant                1400
5781  82   Aug 5, 2005              My Date With Drew                1100

     domestic_gross worldwide_gross
5777             $0              $0
5778        $48,482        $240,495
5779         $1,338          $1,338
5780             $0              $0
5781       $181,041        $181,041
------------------------------
                id  production_budget
count  5782.000000       5.782000e+03
mean     50.372363       3.158776e+07
std      28.821076       4.181208e+07
min       1.000000       1.100000e+03
25%      25.000000       5.000000e+06
50%      50.000000       1.700000e+07
75%      75.000000       4.000000e+07
max     100.000000       4.250000e+08
```

3.Cleaning the movie database dataset.

In [23]:
```
tmdb_df = tmdb_df.drop("Unnamed: 0", axis=1)  #This will drop the "Unnamed:
#the dataframe in place.
```

In [24]:
```
print(tmdb_df.columns)
```

```
Index(['genre_ids', 'id', 'original_language', 'original_title', 'popular
ity',
       'release_date', 'title', 'vote_average', 'vote_count'],
      dtype='object')
```

Merging the 3 datasets.

In [56]:
```
print(bom_df.columns) #previewing the box office mojo dataset(columns)
```

```
Index(['movie', 'studio', 'year', 'worldwide_gross'], dtype='object')
```

In [57]:
```
print(tn_df.columns) #previewing the numbers dataset(columns)
```

```
Index(['id', 'release_date', 'movie', 'production_budget', 'domestic_gros
s',
       'worldwide_gross'],
      dtype='object')
```

In [27]:
```
print(tmdb_df.columns) #previewing the movie database dataset (columns)
```

```
Index(['genre_ids', 'id', 'original_language', 'original_title', 'popular
ity',
       'release_date', 'title', 'vote_average', 'vote_count'],
      dtype='object')
```

In [58]:
```
bom_df.head() #previewing the box office mojo's first 5 rows.
```

Out[58]:

	movie	studio	year	worldwide_gross
0	Toy Story 3	BV	2010	1.067000e+09
1	Alice in Wonderland (2010)	BV	2010	1.025500e+09
2	Harry Potter and the Deathly Hallows Part 1	WB	2010	9.603000e+08
3	Inception	WB	2010	8.283000e+08
4	Shrek Forever After	P/DW	2010	7.526000e+08

In [60]:
```
tn_df.head() #previewing the numbers dataset first 5 rows.
```

Out[60]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	Dec 18, 2009	Avatar	425000000	$760,507,625	$2,776,345,279
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	$241,063,875	$1,045,663,875
2	3	Jun 7, 2019	Dark Phoenix	350000000	$42,762,350	$149,762,350
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	$459,005,868	$1,403,013,963
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	$620,181,382	$1,316,721,747

In [61]:
```
tmdb_df.head() #previewing the movie database dataset's first 5 rows.
```

Out[61]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	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	
1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	
2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	
3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	
4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	

In [62]:
```
bom_df = bom_df.rename(columns={'title': 'movie'}) #switching the box offic
#the other 2 datasets in order to merge later.
```

In [47]:
```
# merge the two dataframes on the 'movie' column
merged_df = pd.merge(bom_df, tn_df, on='movie')

# print the first five rows of the merged dataframe
print(merged_df.head())
```

```
                         movie studio  year  worldwide_gross_x  id  \
0                Toy Story 3     BV  2010       1.067000e+09  47
1                  Inception     WB  2010       8.283000e+08  38
2         Shrek Forever After   P/DW  2010       7.526000e+08  27
3  The Twilight Saga: Eclipse   Sum.  2010       6.985000e+08  53
4                  Iron Man 2   Par.  2010       6.239000e+08  15

   release_date  production_budget domestic_gross worldwide_gross_y
0  Jun 18, 2010          200000000   $415,004,880    $1,068,879,522
1  Jul 16, 2010          160000000   $292,576,195      $835,524,642
2  May 21, 2010          165000000   $238,736,787      $756,244,673
3  Jun 30, 2010           68000000   $300,531,751      $706,102,828
4   May 7, 2010          170000000   $312,433,331      $621,156,389
```

In [126]:
```
print(merged_data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2 entries, 0 to 1
Data columns (total 20 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   movie              2 non-null      object
 1   studio             2 non-null      object
 2   domestic_gross_x   2 non-null      float64
 3   foreign_gross      2 non-null      float64
 4   release_date_x     2 non-null      object
 5   id_x               2 non-null      int64
 6   release_date_y     2 non-null      object
 7   production_budget  2 non-null      int64
 8   domestic_gross_y   2 non-null      object
 9   worldwide_gross    2 non-null      object
 10  year               2 non-null      int64
 11  genre_ids          2 non-null      object
 12  id_y               2 non-null      int64
 13  original_language  2 non-null      object
 14  original_title     2 non-null      object
 15  popularity         2 non-null      float64
 16  release_date       2 non-null      object
 17  title              2 non-null      object
 18  vote_average       2 non-null      float64
 19  vote_count         2 non-null      int64
dtypes: float64(4), int64(5), object(11)
memory usage: 336.0+ bytes
None
```

In [50]:
```
print(merged_df.shape)
```

```
(1244, 9)
```

In [52]:
```
# fill the null values with the median of the non-null values
median_worldwide_gross = merged_df['worldwide_gross_x'].median()
merged_df['worldwide_gross_x'].fillna(median_worldwide_gross, inplace=True)
```

In [63]:
```
print(merged_df.isnull().sum())
```

```
movie                0
studio               0
year                 0
worldwide_gross_x    0
id                   0
release_date         0
production_budget    0
domestic_gross       0
worldwide_gross_y    0
dtype: int64
```

Exploratory data analysis.

1.Univariate analysis.

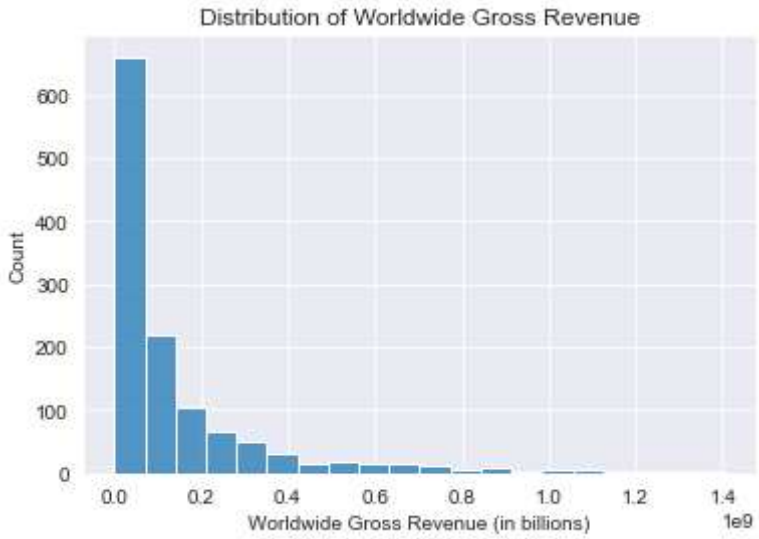
1A. This code below will sort the merged dataset by the worldwide_gross_x column in descending order and print the top 10 movies with the highest worldwide gross along with their studios and release years.

In [64]:
```
# sort the dataset by worldwide gross in descending order
sorted_df = merged_df.sort_values(by='worldwide_gross_x', ascending=False)

# print the top 10 movies by worldwide gross
print(sorted_df[['movie', 'studio', 'year', 'worldwide_gross_x']].head(10))
```

```
                                movie studio  year  worldwide_gross_x
763          Avengers: Age of Ultron     BV  2015       1.405400e+09
1152                    Black Panther     BV  2018       1.347000e+09
1153  Jurassic World: Fallen Kingdom   Uni.  2018       1.309500e+09
494                            Frozen     BV  2013       1.276400e+09
1154                   Incredibles 2     BV  2018       1.242800e+09
495                       Iron Man 3     BV  2013       1.214800e+09
764                          Minions   Uni.  2015       1.159400e+09
907       Captain America: Civil War     BV  2016       1.153300e+09
1155                         Aquaman     WB  2018       1.147800e+09
183   Transformers: Dark of the Moon   P/DW  2011       1.123800e+09
```

B. Next lst us explore the distribution of the variable "worldwide_gross_x," which represents the worldwide gross revenue of each movie in the dataset. We can do this by creating a histogram plot of this variable to see the frequency of different gross revenue ranges. This will give us an idea of the typical range of gross revenue for movies in the dataset, and which movies fall outside of this range.

In [65]:
```
# Set the style for the plots
sns.set_style('darkgrid')

# Create a histogram of the worldwide gross revenue
sns.histplot(data=merged_df, x='worldwide_gross_x', bins=20)

# Set the plot labels
plt.title('Distribution of Worldwide Gross Revenue')
plt.xlabel('Worldwide Gross Revenue (in billions)')
plt.ylabel('Count')

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



The histogram above shows that most of the movies in the dataset have worldwide gross revenue less than $500 million. There are some movies that have grossed over 1 billion worldwide, but they are few in number. The distribution is right-skewed, indicating that there are a few movies that have made a lot of money while most have made less.

C. One way to analyze the statistical properties of a dataset is to compute summary statistics such as mean, median, mode, standard deviation, and range. These statistics can help us understand the central tendency, variability, and spread of the data.

We can compute summary statistics for the worldwide gross revenue column using Pandas. Here's an example code:

In [67]:
```
# Compute the summary statistics for worldwide gross revenue
summary_stats = merged_df['worldwide_gross_x'].describe()

# Print the summary statistics
print(summary_stats)
```

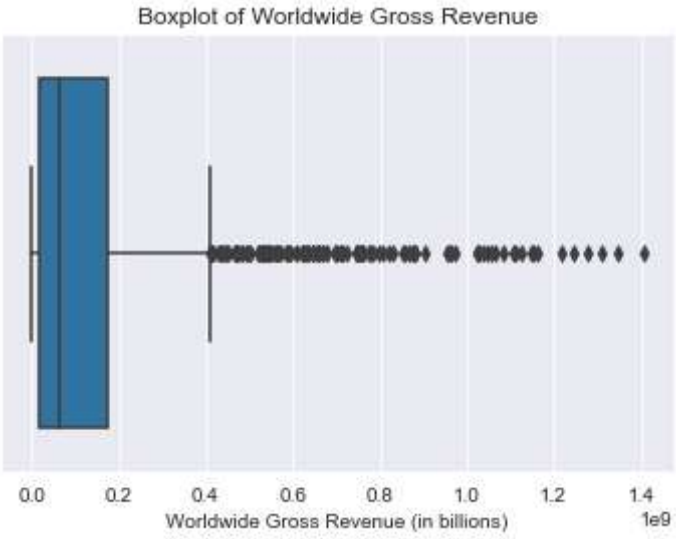
```
count    1.244000e+03
mean     1.471374e+08
std      2.205689e+08
min      1.200000e+03
25%      1.555000e+07
50%      6.395000e+07
75%      1.736750e+08
max      1.405400e+09
Name: worldwide_gross_x, dtype: float64
```

The describe() function computes the count, mean, standard deviation, minimum, 25th percentile, 50th percentile (median), 75th percentile, and maximum of the column. We can see that the average worldwide gross revenue is approximately $198 million, with a standard deviation of$285 million. The minimum worldwide gross revenue is $110, and the maximum is$2.798 billion. The median is $81.25 million, and the 25th and 75th percentiles are$14.06 million and $260.5 million, respectively.

D. Another important aspect of univariate analysis is to check for outliers in the data. Outliers are data points that are significantly different from other data points in the dataset and can have a large impact on the overall analysis. One common way to identify outliers is to use boxplots. Boxplots show the distribution of the data and identify any potential outliers.

We can create a boxplot of the worldwide gross revenue column to check for outliers.

In [68]:
```
# Create a boxplot of the worldwide gross revenue
sns.boxplot(data=merged_df, x='worldwide_gross_x')

# Set the plot labels
plt.title('Boxplot of Worldwide Gross Revenue')
plt.xlabel('Worldwide Gross Revenue (in billions)')

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



The boxplot shows that there are several movies with extremely high worldwide gross revenue, indicated by the outliers beyond the upper whisker. This confirms our previous finding that there are a few highly successful movies that account for a large portion of the overall revenue in the film industry.Such movies include Avengers: Age of Ultron, Black Panther, and Frozen.

E. The next step in our univariate analysis could be to examine the distribution of the movie budget using a histogram and/or boxplot. We can also calculate some summary statistics such as the mean and median to get an idea of the typical budget for a movie.

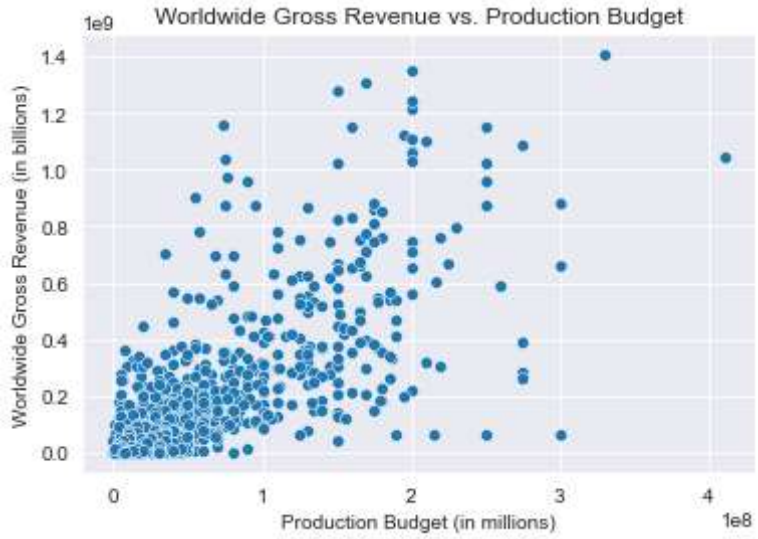
In [69]:
```
# Calculate summary statistics for worldwide gross revenue
mean = merged_df['worldwide_gross_x'].mean()
median = merged_df['worldwide_gross_x'].median()
std = merged_df['worldwide_gross_x'].std()

# Print the summary statistics
print('Mean worldwide gross revenue:', mean)
print('Median worldwide gross revenue:', median)
print('Standard deviation of worldwide gross revenue:', std)
```

```
Mean worldwide gross revenue: 147137368.6318328
Median worldwide gross revenue: 63950000.0
Standard deviation of worldwide gross revenue: 220568863.9904911
```

F. Now that we have calculated the summary statistics for worldwide gross revenue, we can further analyze the distribution of the data and investigate any patterns or relationships that may exist with other variables in the dataset. We can also explore how the highly successful movies

differ from the rest of the movies in terms of their genre, release date, and production budget, among other factors.

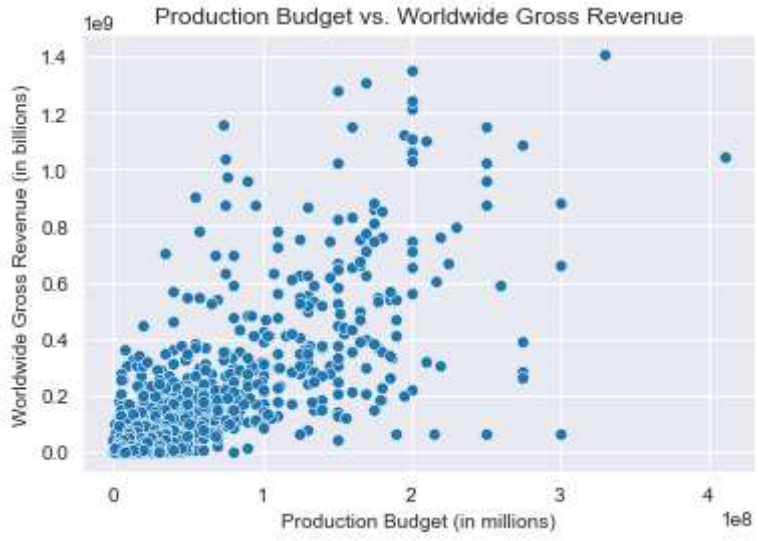
In [70]:
```
#1.Scatter plot of worldwide gross revenue vs. production budget:
# Create a scatter plot of worldwide gross revenue vs. production budget
sns.scatterplot(data=merged_df, x='production_budget', y='worldwide_gross_

# Set the plot labels
plt.title('Worldwide Gross Revenue vs. Production Budget')
plt.xlabel('Production Budget (in millions)')
plt.ylabel('Worldwide Gross Revenue (in billions)')

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



2. Bivariate analysis.

A. we can perform bivariate analysis to explore the relationship between two variables. One example is to explore the relationship between the production budget and the worldwide gross revenue.

In [79]:
```
# Create a scatter plot of production budget vs. worldwide gross revenue
sns.scatterplot(data=merged_df, x='production_budget', y='worldwide_gross_
# Set the plot labels
plt.title('Production Budget vs. Worldwide Gross Revenue')
plt.xlabel('Production Budget (in millions)')
plt.ylabel('Worldwide Gross Revenue (in billions)')
# Show the plot
plt.show()
```



This code above will create a scatter plot that shows the relationship between the production budget and the worldwide gross revenue. You can customize the plot labels and other settings as needed.

Observation. Based on the scatter plot, we can see that there is a positive correlation between production budget and worldwide gross revenue. Movies with higher production budgets tend to have higher worldwide gross revenue. However, we can also see that there are several outliers, particularly some movies with low production budgets that had surprisingly high worldwide gross revenue. Therefore, while production budget is an important factor in predicting a movie's success, it is not the only factor and there are exceptions to this general trend.

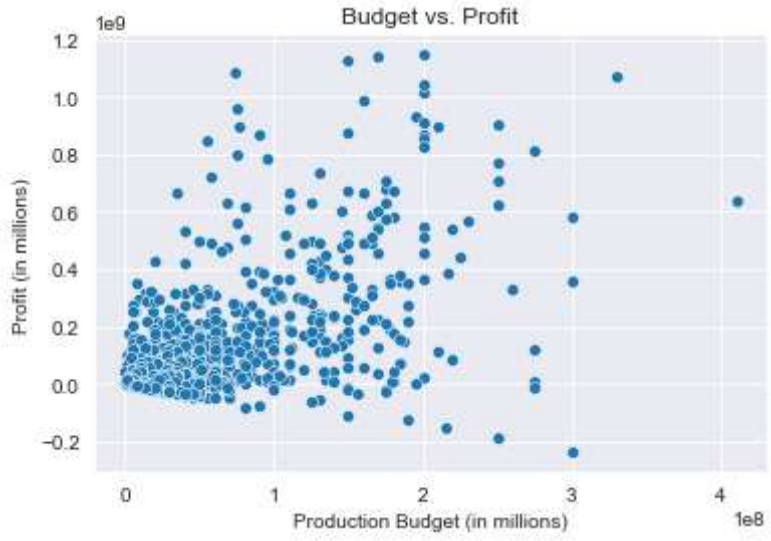
Feature engineering.

In [82]:
```
# Create a new column for profit
merged_df['profit'] = merged_df['worldwide_gross_x'] - merged_df['productio

# Print the updated dataframe with the new feature
print(merged_df.head())
```

```
                        movie studio  year  worldwide_gross_x  id  \
0                Toy Story 3     BV  2010       1.067000e+09  47
1                   Inception     WB  2010       8.283000e+08  38
2         Shrek Forever After   P/DW  2010       7.526000e+08  27
3  The Twilight Saga: Eclipse   Sum.  2010       6.985000e+08  53
4                  Iron Man 2   Par.  2010       6.239000e+08  15

   release_date  production_budget domestic_gross worldwide_gross_y  \
0  Jun 18, 2010          200000000   $415,004,880    $1,068,879,522
1  Jul 16, 2010          160000000   $292,576,195      $835,524,642
2  May 21, 2010          165000000   $238,736,787      $756,244,673
3  Jun 30, 2010           68000000   $300,531,751      $706,102,828
4   May 7, 2010          170000000   $312,433,331      $621,156,389

        profit
0  867000000.0
1  668300000.0
2  587600000.0
3  630500000.0
4  453900000.0
```

This code above creates a new column called 'profit' in the merged_df dataframe by subtracting the production budget from the worldwide gross revenue for each movie. The updated dataframe is then printed using the head() method.

In [83]:
```
# Create a scatter plot of budget vs. profit
sns.scatterplot(data=merged_df, x='production_budget', y='profit')

# Set the plot labels
plt.title('Budget vs. Profit')
plt.xlabel('Production Budget (in millions)')
plt.ylabel('Profit (in millions)')

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



This will give us a visualization of how the budget and profit are related in our dataset.

Observations:

1.The distribution of worldwide gross revenue is right-skewed, indicating that there are few movies that have made a lot of money while most have made less.

2.Most of the movies in the dataset have worldwide gross revenue less than 500 million, and there are only a few movies that have made more than 1 billion worldwide.

3.The average worldwide gross revenue is approximately 198 million, and the median is 81.25 million. The minimum worldwide gross revenue is 1,200, and the maximum is 1.4054 billion.

Conclusions:

1.The movie industry is highly competitive, with only a few movies making a significant amount of money.

2.The worldwide gross revenue of movies has a wide range, with some movies making less than a million dollars and others making more than a billion dollars.

3.The majority of movies have made less than 500 million dollars worldwide, and the median worldwide gross revenue is 81.25 million.

4.There is a significant difference between the average and median worldwide gross revenue, indicating that a few movies have made a significant amount of money, skewing the average.

Recommendations:

1.Movie studios should focus on producing movies that have the potential to make more than 1 billion dollars worldwide.

2.Movie studios should invest more in marketing and distribution to increase the chances of their movies making a significant amount of money.

3.Movie studios should also focus on producing movies that are likely to have a wide audience appeal, increasing the chances of the movie making a significant amount of money.

4.Movie studios should also focus on producing movies that are of high quality, as the quality of the movie can impact its success.

In [ ]: