Project 1

June 12, 2023

1 Business understanding

1.1 Real world problem: How does Microsoft studio maximize Worldwide Gross?

Who are the stakeholders? Microsoft CEO, board and shareholders, project managers on team for Microsoft Studio, CFO.

This notebook should show the value of genre, runtime and budget on the worldwide gross.

[1]:
```
#importing pandas, numpy, matplotlib, seaborn, glob, sql and imdb, budget␣
 ↪datasets
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from glob import glob
import os
from matplotlib.ticker import MultipleLocator
from matplotlib.ticker import FuncFormatter

import sqlite3
connection = sqlite3.connect("imdb")

%matplotlib inline


df = pd.read_csv('tn.movie_budgets.csv')
```

2 Reviewing the data from the tn.movie_budget.csv

Important data used for this project comes from the Movie, production_budget, and worldwide_gross columns

[2]:
```
#displaying data from tn.movie_budgets
# This dataset displays the release date, movie name, production budget and the␣
 ↪domestic, ww gross. This project heavily was influenced on the budget and␣
 ↪gross.
```

```
print(df.head())
```

```
   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
```

[3]:
```
#displaying the diffrent columns in the tn.movie_budgets/info about the dataset
for c in df.columns:
    print(c)
```

```
id
release_date
movie
production_budget
domestic_gross
worldwide_gross
```

3 Reviewing data

There are no missing values. There are 5 object types and one interger type columns.

[4]:
```
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
```

[5]:
```
df.describe()
```

[5]:
```
                id
count  5782.000000
mean     50.372363
std      28.821076
min       1.000000
25%      25.000000
50%      50.000000
75%      75.000000
max     100.000000
```

4 Displaying column info from IMDB

Using SQL to show IMDB movie basics, directors, known for, movie akas, movie ratings, persons, principal, writers and movie.

Subcategory of movie basics had very important info including runtime and genre. These were used to make dec of what type of movie should be produced.

[6]:
```
cursor = connection.cursor()
sql_query = """SELECT name FROM sqlite_master WHERE type='table';"""
cursor.execute(sql_query)
print(cursor.fetchall())
```

```
[('movie_basics',), ('directors',), ('known_for',), ('movie_akas',),
('movie_ratings',), ('persons',), ('principals',), ('writers',), ('movie',)]
```

[7]:
```
#displaying movie_basics
movie_basics_df = pd.read_sql("""SELECT * FROM movie_basics""",connection)
movie_basics_df.head(100)
```

[7]:

	movie_id	primary_title	original_title	start_year	runtime_minutes
0	tt0063540	Sunghursh	Sunghursh	2013	175.0
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN
4	tt0100275	The Wandering Soap Opera			
..	…	…			
95	tt0429493	The A-Team			
96	tt0430524	The Rescuer			
97	tt0431021	The Possession			
98	tt0432010	The Queen of Sheba Meets the Atom Man			
99	tt0433035	Real Steel			

```
4                    La Telenovela Errante        2017             80.0
..                                     …         …                …
95                             The A-Team        2010            117.0
96                           The Rescuer         2011             84.0
97                         The Possession        2012             92.0
98  The Queen of Sheba Meets the Atom Man        2018            110.0
99                             Real Steel        2011            127.0

                       genres
0          Action,Crime,Drama
1             Biography,Drama
2                       Drama
3               Comedy,Drama
4        Comedy,Drama,Fantasy
..                        …
95  Action,Adventure,Thriller
96               Documentary
97    Horror,Mystery,Thriller
98                      Comedy
99        Action,Drama,Family

[100 rows x 6 columns]
```

5 Cleaning data

Running to see if there are any duplicated items in the data set. I also checked to see if there were any placeholder such as null value. In both cases there was no problematic data.

[8]:
```
#cleaning duplicates
print(movie_basics_df.duplicated().any())
```

```
False
```

[9]:
```
#Checking for Placeholders
movie_basics_df.isin(['?', '#', 'NaN', 'null', 'N/A', '-', 0]).sum()
```

[9]:
```
movie_id           0
primary_title      0
original_title     0
start_year         0
runtime_minutes    0
genres             0
dtype: int64
```

6 Merging two diffrent types of data sets

In order to merge an SQL and PANDAS data set (IMDB, tn.movie_budget.csv), there has to be a similar name. I adjusted primary_title to movie on the IMDB dataset. This will allow for a future merge of the datasets.

[10]:
```
#changing primary title to movie
movie_basics_df.rename(columns={"primary_title": "movie"}, inplace=True)
print(movie_basics_df.head())
```

```
    movie_id                          movie              original_title  \
0  tt0063540                       Sunghursh                  Sunghursh
1  tt0066787  One Day Before the Rainy Season             Ashad Ka Ek Din
2  tt0069049       The Other Side of the Wind  The Other Side of the Wind
3  tt0069204                 Sabse Bada Sukh             Sabse Bada Sukh
4  tt0100275       The Wandering Soap Opera       La Telenovela Errante

   start_year  runtime_minutes                genres
0        2013            175.0    Action,Crime,Drama
1        2019            114.0       Biography,Drama
2        2018            122.0                 Drama
3        2018              NaN          Comedy,Drama
4        2017             80.0  Comedy,Drama,Fantasy
```

7 Combined merge of IMDB and tn.movie_budget

This data set will specfically combine just the movie basics info from the IMDB data set with the full tn.movie_budget dataset.

[11]:
```
#full combined merge of IMDB and tn.movie _budget
imdb_basics = movie_basics_df

full_df = pd.merge(left=df, right=imdb_basics, on="movie")
full_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3815 entries, 0 to 3814
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 3815 non-null   int64
 1   release_date       3815 non-null   object
 2   movie              3815 non-null   object
 3   production_budget  3815 non-null   object
 4   domestic_gross     3815 non-null   object
 5   worldwide_gross    3815 non-null   object
 6   movie_id           3815 non-null   object
 7   original_title     3814 non-null   object
 8   start_year         3815 non-null   int64
```

```
 9   runtime_minutes    3328 non-null   float64
 10  genres             3743 non-null   object
dtypes: float64(1), int64(2), object(8)
memory usage: 357.7+ KB
```

[12]:
```
full_df.head()
```

[12]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	movie_id	original_title	start_year	runtime_minutes	genres
0	1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279	tt1775309	Abatâ	2011	93.0	Horror
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875	tt1298650	Pirates of the Caribbean: On Stranger Tides	2011	136.0	Action,Adventure,Fantasy
2	3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350	tt6565702	Dark Phoenix	2019	113.0	Action,Adventure,Sci-Fi
3	4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963	tt2395427	Avengers: Age of Ultron	2015	141.0	Action,Adventure,Sci-Fi
4	7	Apr 27, 2018	Avengers: Infinity War	$300,000,000	$678,815,482	$2,048,134,200	tt4154756	Avengers: Infinity War	2018	149.0	Action,Adventure,Sci-Fi

8 Compressing data into readable format

I removed data that will not be needed for this project. Data removed includes id, release_date, domestic_gross, movie_id, original_title, and start_year. I then checked to ensure all data was removed.

[13]:
```
#removing unneeded data =orignial_title, start_year, Movie_id, id
full_df.drop(["original_title", "movie_id",␣
 ↪"start_year","id","domestic_gross","release_date"], axis=1, inplace=True)
```

[14]:
```
#checking to see if data was removed
full_df.tail()
```

[14]:
```
                     movie production_budget worldwide_gross  runtime_minutes  \
3810                  Cure           $10,000         $94,596              NaN
3811                  Bang           $10,000            $527              NaN
3812             Newlyweds            $9,000          $4,584             95.0
3813                Red 11            $7,000              $0             77.0
3814  A Plague So Pleasant            $1,400              $0             76.0

                      genres
3810                    None
3811                    None
3812            Comedy,Drama
3813  Horror,Sci-Fi,Thriller
3814   Drama,Horror,Thriller
```

9 Cleaning full dataset

I checked to see if there was any missing data in the combined dataset. There were missing values in the runtime_minutes and genre columns. This data was removed from the cleaned dataset.

[15]:
```
#review for missing data/remove
cleaned_df = full_df.dropna(subset=['production_budget'])
cleaned_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3815 entries, 0 to 3814
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   movie              3815 non-null   object
 1   production_budget  3815 non-null   object
 2   worldwide_gross    3815 non-null   object
 3   runtime_minutes    3328 non-null   float64
 4   genres             3743 non-null   object
dtypes: float64(1), object(4)
memory usage: 178.8+ KB
```

[16]:
```
cleaned_df = full_df.dropna(subset=['production_budget', 'worldwide_gross',␣
 ↪'runtime_minutes', 'genres'])
cleaned_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3309 entries, 0 to 3814
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   movie              3309 non-null   object
 1   production_budget  3309 non-null   object
 2   worldwide_gross    3309 non-null   object
```

```
 3   runtime_minutes    3309 non-null   float64
 4   genres             3309 non-null   object
dtypes: float64(1), object(4)
memory usage: 155.1+ KB
```

[17]:
```
cleaned_df = cleaned_df.dropna(subset=['worldwide_gross'])
cleaned_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3309 entries, 0 to 3814
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   movie              3309 non-null   object
 1   production_budget  3309 non-null   object
 2   worldwide_gross    3309 non-null   object
 3   runtime_minutes    3309 non-null   float64
 4   genres             3309 non-null   object
dtypes: float64(1), object(4)
memory usage: 155.1+ KB
```

[18]:
```
cleaned_df = cleaned_df.dropna(subset=['runtime_minutes'])
cleaned_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3309 entries, 0 to 3814
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   movie              3309 non-null   object
 1   production_budget  3309 non-null   object
 2   worldwide_gross    3309 non-null   object
 3   runtime_minutes    3309 non-null   float64
 4   genres             3309 non-null   object
dtypes: float64(1), object(4)
memory usage: 155.1+ KB
```

10 Adjusting data type for cleaned dataset

In order to fully review the data we had to change the datatype for production_budget, and worldwide_gross. This data had to change from a string to an integer. To do this we had to drop the $.

[19]:
```
# Adjusting type of data for production_budget, worldwide_gross
#From object to int

cleaned_df['production_budget'] = cleaned_df['production_budget'].str.
 ↪replace(',', '')
```

```
cleaned_df['production_budget'] = cleaned_df['production_budget'].str.
 ↪replace('$', '')
cleaned_df['production_budget'] = cleaned_df['production_budget'].astype(int)
```

[20]:
```
cleaned_df.head()
```

[20]:
```
                                    movie  production_budget  \
0                                  Avatar          425000000
1  Pirates of the Caribbean: On Stranger Tides          410600000
2                                Dark Phoenix          350000000
3                     Avengers: Age of Ultron          330600000
4                      Avengers: Infinity War          300000000

  worldwide_gross  runtime_minutes                    genres
0  $2,776,345,279             93.0                    Horror
1  $1,045,663,875            136.0  Action,Adventure,Fantasy
2    $149,762,350            113.0   Action,Adventure,Sci-Fi
3  $1,403,013,963            141.0   Action,Adventure,Sci-Fi
4  $2,048,134,200            149.0   Action,Adventure,Sci-Fi
```

[21]:
```
cleaned_df['worldwide_gross'] = cleaned_df['worldwide_gross'].str.replace(',',␣
 ↪'')
cleaned_df['worldwide_gross'] = cleaned_df['worldwide_gross'].str.replace('$',␣
 ↪'')
cleaned_df['worldwide_gross'] = cleaned_df['worldwide_gross'].astype(int)
```

[22]:
```
cleaned_df.head()
```

[22]:
```
                                    movie  production_budget  \
0                                  Avatar          425000000
1  Pirates of the Caribbean: On Stranger Tides          410600000
2                                Dark Phoenix          350000000
3                     Avengers: Age of Ultron          330600000
4                      Avengers: Infinity War          300000000

   worldwide_gross  runtime_minutes                    genres
0       2776345279             93.0                    Horror
1       1045663875            136.0  Action,Adventure,Fantasy
2        149762350            113.0   Action,Adventure,Sci-Fi
3       1403013963            141.0   Action,Adventure,Sci-Fi
4       2048134200            149.0   Action,Adventure,Sci-Fi
```

11 Reviewing cleaned data

Ensuring there are no missing values. Full count of 3,309 non-null meaning there is no missing data.

[23]:
```
cleaned_df = cleaned_df.dropna(subset=['genres'])
cleaned_df.info()
```

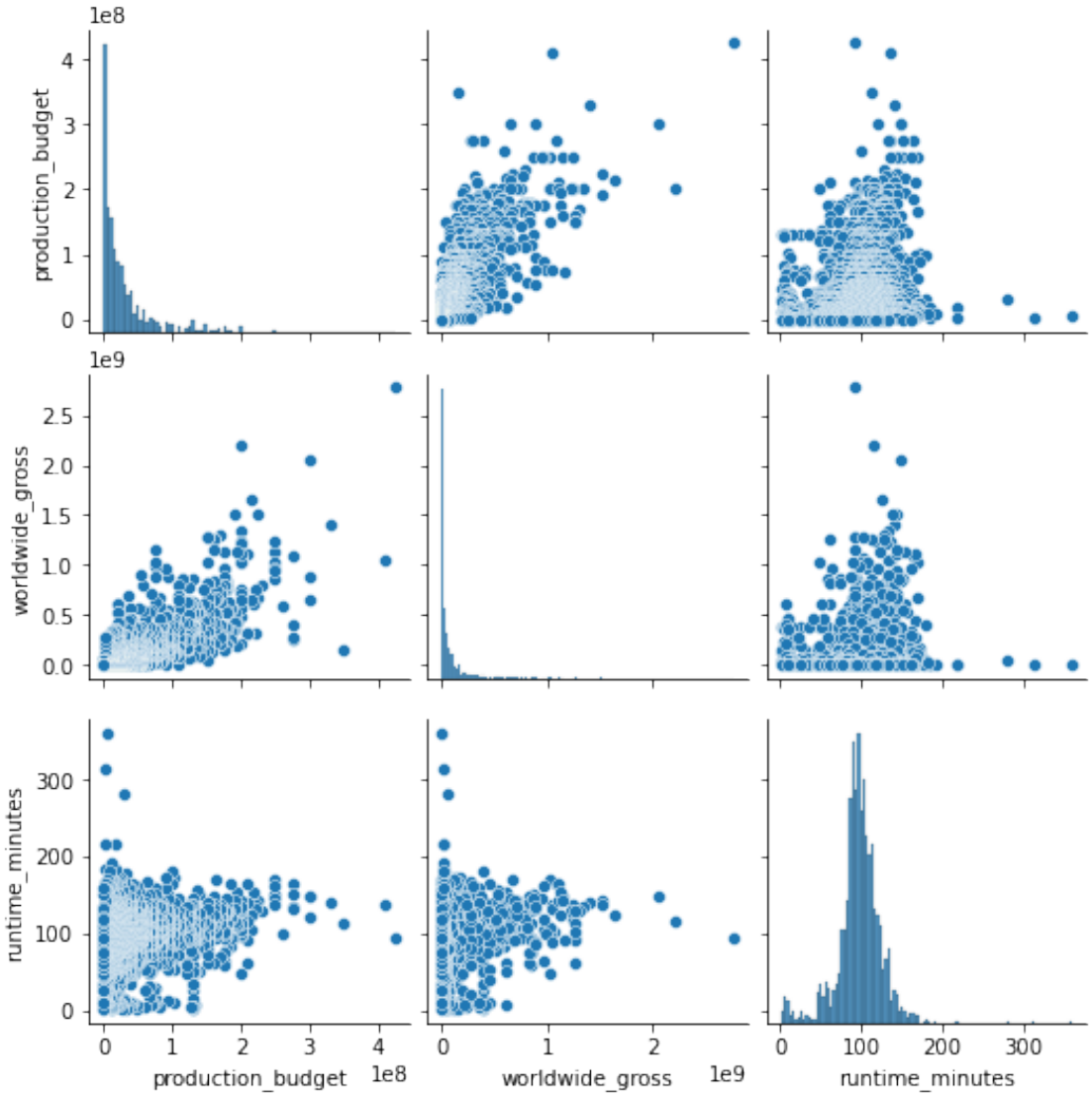
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3309 entries, 0 to 3814
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   movie              3309 non-null   object
 1   production_budget  3309 non-null   int64
 2   worldwide_gross    3309 non-null   int64
 3   runtime_minutes    3309 non-null   float64
 4   genres             3309 non-null   object
dtypes: float64(1), int64(2), object(2)
memory usage: 155.1+ KB
```

12 Plots

These initial plots show the coorelation between production budget, worldwide gorss and runtime.

[24]:
```
#plots
sns.pairplot(cleaned_df)
```

[24]:
```
<seaborn.axisgrid.PairGrid at 0x7f7d818f1280>
```



[25]:
```
#Review columns of cleaned data
c = list(cleaned_df.columns)
c = c[1:]
```

[26]:
```
c
```

[26]: `['production_budget', 'worldwide_gross', 'runtime_minutes', 'genres']`

13 Movie Worldwide Gross data

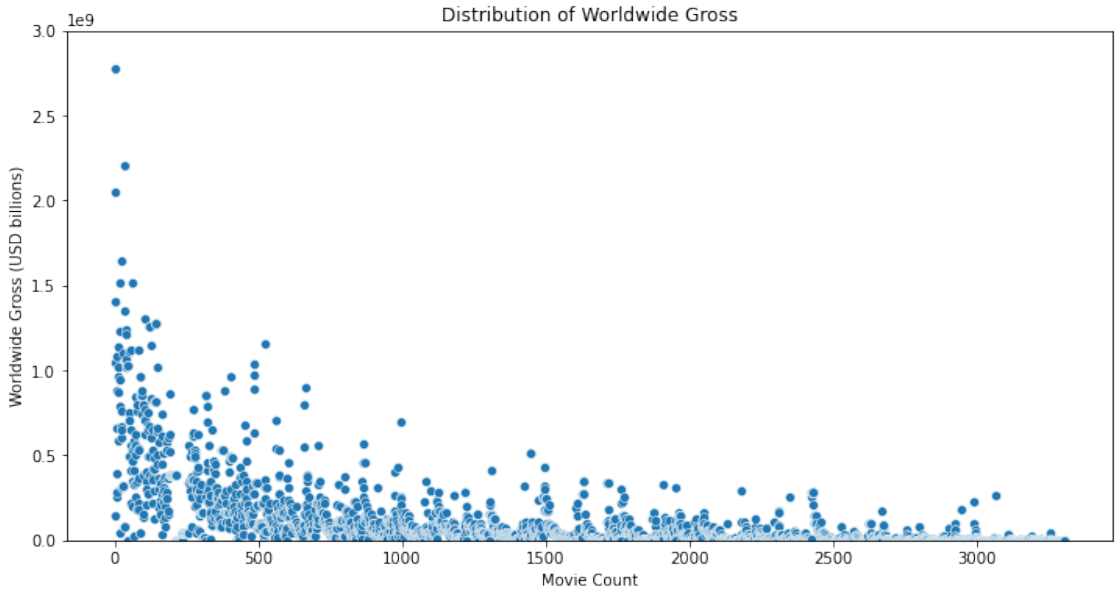
The count indicates there are 3,309 movies in this dataset. The mean or average movie grosses 98,220,550.00. The standard deviaton however shows there is a large spread of data. For instance if we look at the first quartline 25 percent of movies gross below 2,179,623.00. This number is well

below the mean value. Also the largest grossing movie is well above the mean. Avatar grossed over 2.7 Billion dollars. In order to analyze what type of movie grosses well, we must look at other variables.

[27]:
```
cleaned_df['worldwide_gross'].describe()
```

[27]:
```
count    3.309000e+03
mean     9.822055e+07
std      1.965430e+08
min      0.000000e+00
25%      2.179623e+06
50%      2.685381e+07
75%      9.765154e+07
max      2.776345e+09
Name: worldwide_gross, dtype: float64
```

[28]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a scatter plot of worldwide gross
plt.figure(figsize=(12, 6))
sns.scatterplot(data=cleaned_df.reset_index(), x=cleaned_df.reset_index().
 ↪index, y='worldwide_gross')
plt.xlabel('Movie Count')
plt.ylabel('Worldwide Gross (USD billions)')
plt.title('Distribution of Worldwide Gross')
plt.ylim(0, 3e9)  # Set the y-axis limits to 0 to 3 billion
plt.show()
```



[29]:
```
#worldwide gross by budget
total_gross = cleaned_df.sort_values(by="worldwide_gross", ascending=False)
total_gross.head(10)
```

[29]:

	movie	production_budget	worldwide_gross	runtime_minutes	genres
0	Avatar	425000000	2776345279	93.0	Horror
43	Titanic	200000000	2208208395	115.0	Family
4	Avengers: Infinity War	300000000	2048134200	149.0	Action,Adventure,Sci-Fi
25	Jurassic World	215000000	1648854864	124.0	Action,Adventure,Sci-Fi
67	Furious 7	190000000	1518722794	137.0	Action,Crime,Thriller
19	The Avengers	225000000	1517935897	143.0	Action,Adventure,Sci-Fi
3	Avengers: Age of Ultron	330600000	1403013963	141.0	Action,Adventure,Sci-Fi
41	Black Panther	200000000	1348258224	134.0	Action,Adventure,Sci-Fi
116	Jurassic World: Fallen Kingdom	170000000	1305772799	128.0	Action,Adventure,Sci-Fi
156	Frozen	150000000	1272469910	93.0	Adventure,Drama,Sport

[30]:
```
#What is the longest movie?
total_gross["runtime_minutes"].max()
```

[30]:
```
360.0
```

14 List of all the Genres

[31]:
```
#Genres
print(cleaned_df["genres"].unique())
```

```
['Horror' 'Action,Adventure,Fantasy' 'Action,Adventure,Sci-Fi'
 'Action,Adventure,Thriller' 'Action,Thriller' 'Action,Adventure,Western'
 'Adventure,Animation,Comedy' 'Adventure,Family,Fantasy'
 'Adventure,Fantasy' 'Action,Crime,Thriller' 'Action,Adventure,Comedy'
 'Action,Adventure,Drama' 'Action,Drama' 'Action,Adventure,History'
 'Family' 'Action,Adventure,Animation' 'Documentary' 'Fantasy,Musical'
 'Action,Adventure,Horror' 'Drama,Romance' 'Comedy,Drama,Family'
 'Drama,Mystery,Sci-Fi' 'Adventure,Comedy,Family'
 'Action,Adventure,Family' 'Adventure,Drama,Family' 'Action,Horror,Sci-Fi'
```

'Action,Sci-Fi' 'Animation' 'Crime,Drama' 'Biography,Documentary,History'
'Adventure,Drama,Sci-Fi' 'Drama,Fantasy,Romance' 'Family,Fantasy,Musical'
'Action,Drama,History' 'Sci-Fi' 'Documentary,Drama,Sport'
'Adventure,Drama,Sport' 'Fantasy,Romance' 'Action,Drama,Fantasy'
'Comedy,Fantasy,Horror' 'Action,Drama,Thriller' 'Drama,Fantasy,Horror'
'Adventure,Animation,Family' 'Adventure,Animation,Drama'
'Action,Comedy,Fantasy' 'Action,Comedy' 'Action,Adventure,Biography'
'Thriller' 'Drama' 'Documentary,Drama,Family' 'Drama,Horror,Thriller'
'Biography,Documentary' 'Documentary,History' 'Documentary,Family'
'Animation,Documentary' 'Drama,Romance,Thriller' 'Comedy,Family,Fantasy'
'Action,Animation,Comedy' 'Action,Mystery,Thriller' 'Drama,Thriller'
'Animation,Documentary,Family' 'Action,Drama,Sci-Fi'
'Action,Adventure,Crime' 'Adventure,Mystery,Sci-Fi'
'Action,Adventure,Mystery' 'Adventure,Drama,Fantasy'
'Action,Crime,Sci-Fi' 'Documentary,News' 'Documentary,Drama' 'Comedy'
'Comedy,Drama' 'Action,Sci-Fi,Thriller' 'Drama,Sci-Fi,Thriller'
'Action,Fantasy,War' 'Drama,Romance,Sci-Fi' 'Action,Comedy,Crime'
'Action,Drama,Family' 'Adventure,Sport' 'Action,Drama,Romance'
'Documentary,Thriller' 'Action,Drama,Mystery' 'Drama,Western'
'Comedy,Romance' 'Biography,Crime,Drama' 'Comedy,Documentary,Drama'
'Action,Crime,Sport' 'Horror,Sci-Fi,Thriller' 'Musical'
'Drama,Family,Fantasy' 'Music' 'Fantasy' 'Comedy,Drama,Romance'
'Action,Crime' 'Adventure,Comedy,Drama' 'Adventure' 'Crime,Drama,Mystery'
'Action,Comedy,Sci-Fi' 'Action,Adventure,Romance' 'Drama,History'
'Action,Family,Fantasy' 'Action,Crime,Fantasy' 'Adventure,Drama'
'Animation,Comedy,Family' 'Action,Comedy,Family' 'Drama,Fantasy,Musical'
'Documentary,Sport' 'Action,Comedy,Horror' 'Biography,Drama,Musical'
'Drama,Horror,Mystery' 'Biography,Documentary,Drama' 'Comedy,Mystery'
'Mystery,Thriller' 'Action,Drama,War' 'Comedy,Family,Romance'
'Horror,Thriller' 'Crime,Horror,Thriller' 'Drama,Mystery,Thriller'
'Romance' 'Action' 'Crime,Mystery,Thriller' 'Biography,Drama'
'Action,Comedy,Romance' 'Action,Crime,Drama' 'Comedy,Drama,History'
'Adventure,Biography,Documentary' 'Biography,Drama,Thriller'
'Drama,History,War' 'Drama,Family' 'Crime,Drama,Thriller'
'Action,Thriller,War' 'Action,Mystery,Sci-Fi' 'Action,Animation,Fantasy'
'Comedy,Drama,Musical' 'Action,Crime,Mystery' 'Crime,Documentary,Drama'
'Drama,Sci-Fi' 'Action,Fantasy,Horror' 'Biography,Crime,Documentary'
'Comedy,Romance,Sci-Fi' 'Biography,Drama,History' 'Crime,Drama,Family'
'Biography,Drama,Sport' 'Adventure,Comedy' 'Biography,Drama,Western'
'Comedy,Crime,Drama' 'Mystery' 'Biography,Drama,Romance' 'Crime,Thriller'
'Biography,Comedy,Drama' 'Action,Romance' 'Drama,Sport' 'Comedy,Western'
'Action,Biography,Drama' 'Action,Fantasy,Thriller'
'Biography,Drama,Music' 'Comedy,Crime,Romance' 'Adventure,Comedy,Crime'
'Drama,Music,Musical' 'Horror,Mystery,Thriller' 'Adventure,Drama,Horror'
'Drama,Musical,Romance' 'Comedy,Fantasy' 'Crime' 'Crime,Drama,Fantasy'
'Romance,Sci-Fi,Thriller' 'Mystery,Sci-Fi,Thriller' 'Action,Comedy,Drama'
'Action,Biography,Comedy' 'Comedy,Family' 'Adventure,Comedy,Fantasy'
'Crime,Drama,History' 'Biography,Documentary,Family'

'Adventure,Drama,Thriller' 'Biography,Documentary,Sport'
'Comedy,Drama,Thriller' 'Documentary,Fantasy,Mystery'
'Adventure,Drama,History' 'Drama,War' 'Comedy,Music' 'Drama,Horror'
'Animation,Family,Fantasy' 'Animation,Drama,Fantasy'
'Action,Comedy,Sport' 'Fantasy,Horror,Mystery' 'Action,Fantasy,Western'
'Comedy,Crime' 'Drama,History,Thriller' 'Adventure,Comedy,Music'
'Horror,Mystery' 'Adventure,Comedy,Sci-Fi' 'War' 'Comedy,Drama,Sport'
'Comedy,Horror,Thriller' 'Biography,Drama,Family' 'Drama,Horror,Sci-Fi'
'Drama,Family,Sport' 'Drama,Fantasy,Thriller' 'Action,Documentary'
'Horror,Mystery,Sci-Fi' 'Action,History' 'Drama,Family,Romance'
'Drama,Music' 'Documentary,Music' 'Comedy,Sci-Fi' 'Biography,Drama,War'
'Adventure,Biography,Drama' 'Adventure,Drama,Western'
'Adventure,Documentary,Drama' 'Crime,Drama,Horror' 'Sci-Fi,Thriller'
'Drama,Fantasy' 'Animation,Comedy,Drama' 'Comedy,Romance,Sport'
'Documentary,Drama,War' 'Adventure,Comedy,Romance' 'Drama,Music,Romance'
'Comedy,Horror,Sci-Fi' 'Adventure,Documentary'
'Biography,Documentary,Music' 'Biography,Comedy,Documentary'
'Comedy,Horror,Romance' 'Comedy,Fantasy,Romance' 'Comedy,Crime,Thriller'
'Comedy,Horror' 'Adventure,Animation,Documentary'
'Adventure,Drama,Romance' 'Action,Sport' 'Action,Biography,Crime'
'Drama,History,Sport' 'Animation,Sci-Fi' 'Adventure,Documentary,Music'
'Comedy,Drama,Music' 'Action,Comedy,Mystery'
'Action,Adventure,Documentary' 'Action,Drama,Sport'
'Adventure,Fantasy,Mystery' 'Drama,Romance,War' 'Action,Horror,Mystery'
'Fantasy,Horror,Thriller' 'Action,Romance,Sport' 'Drama,Mystery'
'Adventure,Documentary,War' 'Drama,History,Romance'
'Action,Drama,Western' 'Biography,Drama,Mystery' 'Drama,Mystery,Romance'
'Adventure,Documentary,Sport' 'Drama,Music,Thriller'
'Family,Horror,Romance' 'Action,Romance,Thriller' 'Drama,Fantasy,Mystery'
'Comedy,Music,Romance' 'Drama,Fantasy,Sci-Fi' 'Action,Comedy,Documentary'
'Crime,Horror,Mystery' 'Documentary,Western' 'Comedy,Mystery,Thriller'
'Comedy,Drama,Fantasy' 'Sport' 'Crime,Fantasy,Thriller' 'Comedy,Sport'
'Adventure,Documentary,History' 'Drama,Family,Music' 'Documentary,War'
'Comedy,Mystery,Sci-Fi' 'Comedy,Documentary' 'Animation,Drama'
'Adventure,Biography,Comedy' 'Horror,Music,Thriller' 'Comedy,Music,War'
'Documentary,History,Western' 'Adventure,Animation' 'Action,Adventure'
'Action,Documentary,Drama' 'Adventure,Family,Sci-Fi' 'Drama,Thriller,War'
'Biography,Comedy,Crime' 'Western' 'Drama,Mystery,War'
'Comedy,Drama,Mystery' 'Documentary,Drama,News' 'Adventure,Crime,Drama'
'Family,Fantasy,Music' 'Drama,Fantasy,Music' 'Adventure,Horror,Sci-Fi'
'Adventure,Comedy,Horror' 'Action,Horror,Thriller' 'Biography'
'Drama,History,Mystery' 'Comedy,Horror,Mystery' 'Action,Horror'
'Biography,Family,Sport' 'Comedy,Drama,Horror' 'Drama,Family,Thriller'
'Comedy,Thriller' 'Adventure,Family' 'Crime,Mystery,Sci-Fi'
'Documentary,Sport,Thriller' 'Drama,Family,History'
'Fantasy,Horror,Sci-Fi' 'Adventure,Fantasy,Horror'
'Adventure,Horror,Mystery' 'Animation,Documentary,Sci-Fi'
'Animation,Horror' 'Crime,Drama,Romance' 'Drama,Musical'

```
'Animation,Family' 'Drama,Family,Mystery' 'Action,Crime,Horror'
'Adventure,Crime,Thriller' 'Horror,Romance,Thriller'
'Biography,Drama,Fantasy' 'Comedy,Fantasy,Sci-Fi'
'Crime,Documentary,History' 'Fantasy,Horror' 'Drama,Thriller,Western'
'Crime,Documentary' 'Comedy,Fantasy,Musical' 'Documentary,Drama,History'
'Horror,Sci-Fi' 'Documentary,History,War' 'Biography,Documentary,War'
'Comedy,Romance,Thriller' 'Comedy,Crime,Horror' 'Adventure,Horror'
'Music,War' 'Documentary,Music,War' 'Documentary,Drama,Reality-TV'
'Comedy,Fantasy,Thriller' 'Documentary,Horror'
'Documentary,Family,History' 'Adventure,Drama,Mystery'
'Action,Biography,Documentary' 'Horror,Musical' 'Family,Sci-Fi']
```

15 How many Genres are there?

[32]:
```
# number of genres

print(cleaned_df["genres"].nunique())
```

```
335
```

16 Creating a list for Generes

This seprates the total number of genres and creates a list. Some movies may have more than one type of genre. (Ex Jurassic World is Action, Adventure and Sci-Fi).

[33]:
```
#Reviewing genres and relation to gross
total_gross["genres"] = total_gross["genres"].apply(lambda x: x.split(",") if x␣
 ↪else x)
total_gross.tail()
```

[33]:

	movie	production_budget	worldwide_gross	runtime_minutes	genres
2950	Hansel & Gretel Get Baked	4500000	0	86.0	[Comedy, Horror]
2951	Fugly	4500000	0	134.0	[Drama, Thriller]
2952	Zipper	4500000	0	112.0	[Drama, Thriller]
2953	The Final Girls	4500000	0	88.0	[Comedy, Drama, Fantasy]
3814	A Plague So Pleasant	1400	0	76.0	[Drama, Horror, Thriller]

[34]:
```
all_genres = set()

for genres in total_gross["genres"]:
```

```
    if genres:
        all_genres.update(genres)

all_genres
```

[34]:
```
{'Action',
 'Adventure',
 'Animation',
 'Biography',
 'Comedy',
 'Crime',
 'Documentary',
 'Drama',
 'Family',
 'Fantasy',
 'History',
 'Horror',
 'Music',
 'Musical',
 'Mystery',
 'News',
 'Reality-TV',
 'Romance',
 'Sci-Fi',
 'Sport',
 'Thriller',
 'War',
 'Western'}
```

17 Tallying individual generes by movie

The next two steps were used to show all possible genres on the table for the movie. Followed by adding a 1.00 if the movie has that specific genre. This allows for futher analyzation of individual genres and their impact on worldwide_gross.

[35]:
```
# Iterate through the set.
for genre in all_genres:
    # Make a new column in dataframe and fill the columns with zeros.
    total_gross[genre] = np.zeros(shape=total_gross.shape[0])
# Check that the changes took place.
total_gross.head()
```

[35]:

	movie	production_budget	worldwide_gross	\
0	Avatar	425000000	2776345279	
43	Titanic	200000000	2208208395	
4	Avengers: Infinity War	300000000	2048134200	
25	Jurassic World	215000000	1648854864	

```
67              Furious 7          190000000       1518722794

    runtime_minutes                        genres  Animation  Action  \
0              93.0                      [Horror]        0.0     0.0
43            115.0                      [Family]        0.0     0.0
4             149.0  [Action, Adventure, Sci-Fi]        0.0     0.0
25            124.0  [Action, Adventure, Sci-Fi]        0.0     0.0
67            137.0    [Action, Crime, Thriller]        0.0     0.0

    Reality-TV  War  Family  …  Music  Sport  History  Mystery  Thriller  \
0          0.0  0.0     0.0  …    0.0    0.0      0.0      0.0       0.0
43         0.0  0.0     0.0  …    0.0    0.0      0.0      0.0       0.0
4          0.0  0.0     0.0  …    0.0    0.0      0.0      0.0       0.0
25         0.0  0.0     0.0  …    0.0    0.0      0.0      0.0       0.0
67         0.0  0.0     0.0  …    0.0    0.0      0.0      0.0       0.0

    Fantasy  News  Sci-Fi  Horror  Adventure
0       0.0   0.0     0.0     0.0        0.0
43      0.0   0.0     0.0     0.0        0.0
4       0.0   0.0     0.0     0.0        0.0
25      0.0   0.0     0.0     0.0        0.0
67      0.0   0.0     0.0     0.0        0.0

[5 rows x 28 columns]
```

[36]:
```
# Iterate through the genres column as an index. Movies can have more than one genre this should count all genres
# the genre rows.
for index, row in total_gross.iterrows():
    # If the value in genres
    if row['genres']:
        # mathces a genre column
        for genre in row['genres']:
            # change that value to 1.
            total_gross.loc[index, genre] = 1
# Lets check our changes.
total_gross.head()
```

[36]:
```
                    movie  production_budget  worldwide_gross  \
0                  Avatar          425000000       2776345279
43                Titanic          200000000       2208208395
4   Avengers: Infinity War          300000000       2048134200
25         Jurassic World          215000000       1648854864
67              Furious 7          190000000       1518722794

    runtime_minutes                        genres  Animation  Action  \
0              93.0                      [Horror]        0.0     0.0
```

```
43            115.0                     [Family]        0.0     0.0
4             149.0  [Action, Adventure, Sci-Fi]        0.0     1.0
25            124.0  [Action, Adventure, Sci-Fi]        0.0     1.0
67            137.0    [Action, Crime, Thriller]        0.0     1.0

    Reality-TV  War  Family  …  Music  Sport  History  Mystery  Thriller  \
0          0.0  0.0     0.0  …    0.0    0.0      0.0      0.0       0.0
43         0.0  0.0     1.0  …    0.0    0.0      0.0      0.0       0.0
4          0.0  0.0     0.0  …    0.0    0.0      0.0      0.0       0.0
25         0.0  0.0     0.0  …    0.0    0.0      0.0      0.0       0.0
67         0.0  0.0     0.0  …    0.0    0.0      0.0      0.0       1.0

    Fantasy  News  Sci-Fi  Horror  Adventure
0       0.0   0.0     0.0     1.0        0.0
43      0.0   0.0     0.0     0.0        0.0
4       0.0   0.0     1.0     0.0        1.0
25      0.0   0.0     1.0     0.0        1.0
67      0.0   0.0     0.0     0.0        0.0

[5 rows x 28 columns]
```

[37]:
```
c = list(total_gross.columns)
```

[38]:
```
genre_c = c[8:]
```

[39]:
```
genre_count = {}
for c in genre_c:
    count = np.sum(total_gross[c] == 1).sum()
    genre_count[c] = count
```

18 How many movies are there in each genre?

This code shows the amount of movies in each genre. Note that a movie can have more than one genre.

[40]:
```

genre_count
```

[40]:
```
{'War': 48,
 'Family': 189,
 'Musical': 28,
 'Biography': 247,
 'Romance': 350,
 'Crime': 379,
 'Documentary': 452,
 'Western': 18,
 'Comedy': 793,
```

```
 'Drama': 1641,
 'Music': 84,
 'Sport': 70,
 'History': 90,
 'Mystery': 233,
 'Thriller': 544,
 'Fantasy': 184,
 'News': 7,
 'Sci-Fi': 212,
 'Horror': 381,
 'Adventure': 469}
```

[41]:
```
#cleaning total gross/genre data- has to be added so we can incorporate the new␣
 ↪genre data
cleaned_df_explode = total_gross.explode("genres")
cleaned_df_explode
```

[41]:

	movie	production_budget	worldwide_gross	\
0	Avatar	425000000	2776345279	
43	Titanic	200000000	2208208395	
4	Avengers: Infinity War	300000000	2048134200	
4	Avengers: Infinity War	300000000	2048134200	
4	Avengers: Infinity War	300000000	2048134200	
…	…	…	…	
2953	The Final Girls	4500000	0	
2953	The Final Girls	4500000	0	
3814	A Plague So Pleasant	1400	0	
3814	A Plague So Pleasant	1400	0	
3814	A Plague So Pleasant	1400	0	

	runtime_minutes	genres	Animation	Action	Reality-TV	War	Family	\
0	93.0	Horror	0.0	0.0	0.0	0.0	0.0	
43	115.0	Family	0.0	0.0	0.0	0.0	1.0	
4	149.0	Action	0.0	1.0	0.0	0.0	0.0	
4	149.0	Adventure	0.0	1.0	0.0	0.0	0.0	
4	149.0	Sci-Fi	0.0	1.0	0.0	0.0	0.0	
…	…	…	…	…	…	…	…	
2953	88.0	Drama	0.0	0.0	0.0	0.0	0.0	
2953	88.0	Fantasy	0.0	0.0	0.0	0.0	0.0	
3814	76.0	Drama	0.0	0.0	0.0	0.0	0.0	
3814	76.0	Horror	0.0	0.0	0.0	0.0	0.0	
3814	76.0	Thriller	0.0	0.0	0.0	0.0	0.0	

	…	Music	Sport	History	Mystery	Thriller	Fantasy	News	Sci-Fi	\
0	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
43	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	…	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	

```
4     …    0.0    0.0      0.0      0.0       0.0      0.0   0.0     1.0
4     …    0.0    0.0      0.0      0.0       0.0      0.0   0.0     1.0
…   …    …    …       …       …       …      …   …      …
2953  …    0.0    0.0      0.0      0.0       0.0      1.0   0.0     0.0
2953  …    0.0    0.0      0.0      0.0       0.0      1.0   0.0     0.0
3814  …    0.0    0.0      0.0      0.0       1.0      0.0   0.0     0.0
3814  …    0.0    0.0      0.0      0.0       1.0      0.0   0.0     0.0
3814  …    0.0    0.0      0.0      0.0       1.0      0.0   0.0     0.0

      Horror  Adventure
0        1.0        0.0
43       0.0        0.0
4        0.0        1.0
4        0.0        1.0
4        0.0        1.0
…       …         …
2953     0.0        0.0
2953     0.0        0.0
3814     1.0        0.0
3814     1.0        0.0
3814     1.0        0.0

[7204 rows x 28 columns]
```

19 Statistical measures of worldwide gross for total dataset

Count-unique genre=305 Mean- World wide gross=94,3374,160 Std- 146,364,500.

[42]:
```
ww_gross_genre_df = cleaned_df_explode.groupby('genres')['worldwide_gross'].
  ↪sum() / cleaned_df_explode['genres'].value_counts()
ww_gross_genre_df.describe()
```

[42]:
```
count    2.300000e+01
mean     1.165129e+08
std      8.766519e+07
min      0.000000e+00
25%      6.359949e+07
50%      7.322647e+07
75%      1.740119e+08
max      3.095840e+08
dtype: float64
```

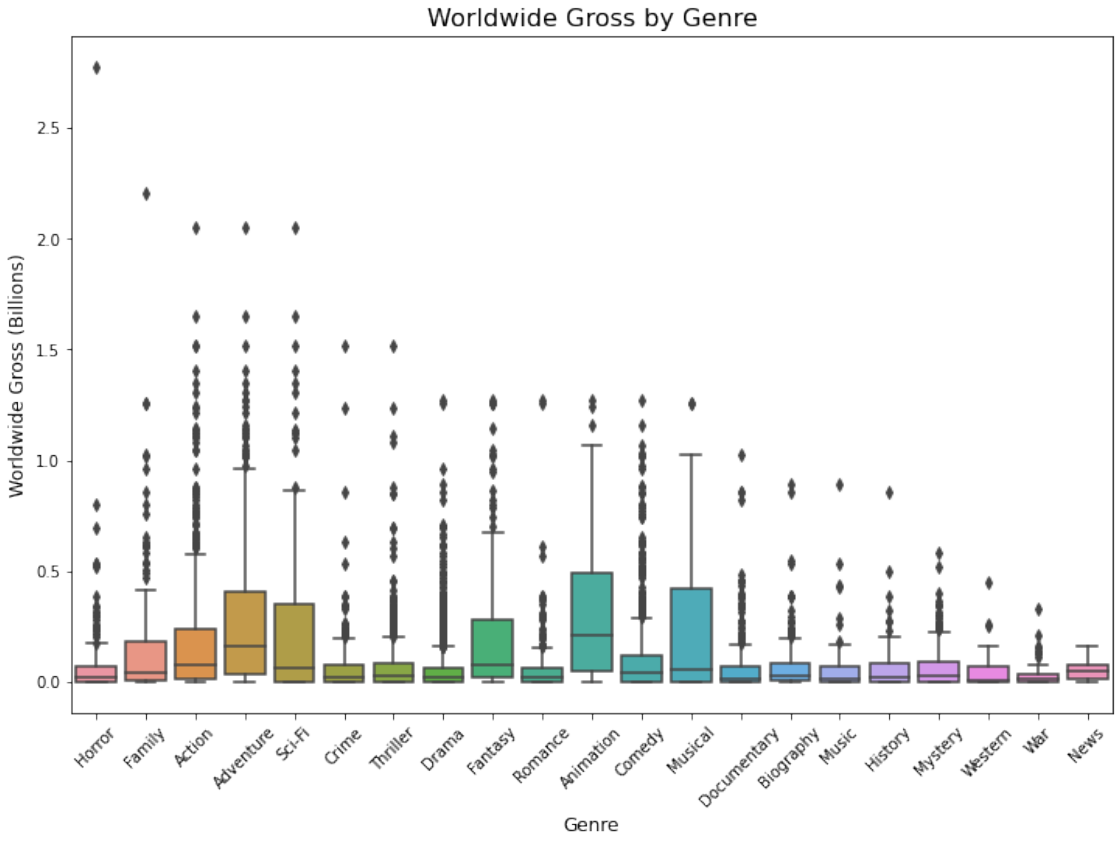
20 Ploting worldwide gross by genre

Movies that produce a high gross(Sci-Fi, Action, Animation, Adventure).

Movies that produce a low gross(News, documentary, Musical, Western, ect).

[43]:
```
#Genres
selected_genres = ['Drama', 'Musical', 'Sci-Fi', 'Western', 'Horror',␣
 ↪'Animation', 'War', 'Comedy',
                   'Fantasy', 'Crime', 'Music', 'Action', 'News', 'Thriller',␣
 ↪'Family', 'Romance',
                   'Biography', 'History', 'Adventure', 'Documentary',␣
 ↪'Mystery']

# Subsetting the data for selected genres
subset_df = cleaned_df_explode[cleaned_df_explode['genres'].
 ↪isin(selected_genres)]

# Converting worldwide gross to billions
subset_df['worldwide_gross'] = subset_df['worldwide_gross'] / 1e9

# Plotting the box plot
plt.figure(figsize=(12, 8))
sns.boxplot(data=subset_df, x='genres', y='worldwide_gross')
plt.title('Worldwide Gross by Genre', fontsize=16)
plt.xlabel('Genre', fontsize=12)
plt.ylabel('Worldwide Gross (Billions)', fontsize=12)
plt.xticks(rotation=45)
plt.show()
```

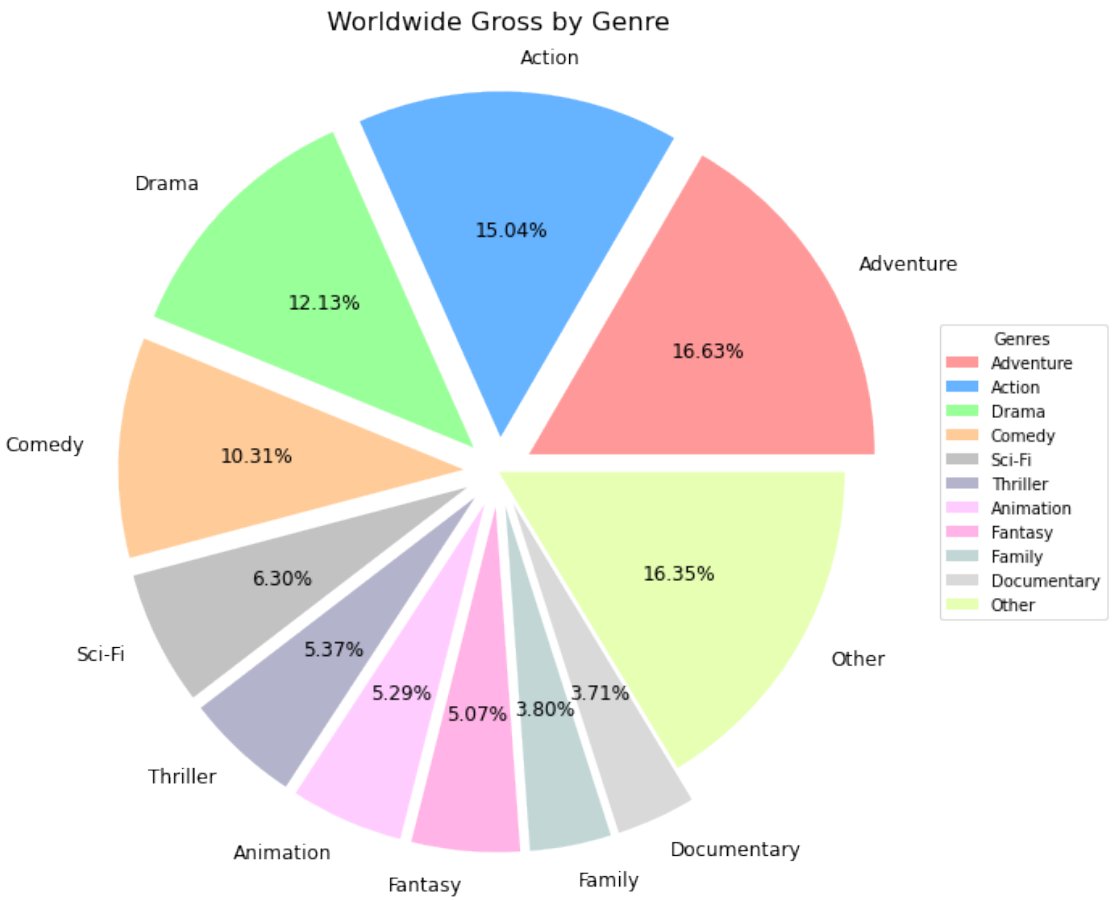
```
<ipython-input-43-475f9af719c1>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  subset_df['worldwide_gross'] = subset_df['worldwide_gross'] / 1e9
```



21 Top 10 Movie Genres for Worldwide Gross return

[44]:
```
# Group the data by genre and calculate the total worldwide gross
genre_gross = cleaned_df_explode.groupby('genres')['worldwide_gross'].sum()

#top 10 genres ww gross
top_genres = genre_gross.nlargest(10)

# Top 11+
other_genre_sum = genre_gross.drop(top_genres.index).sum()

# Top genres with others
combined_genres = top_genres.append(pd.Series(other_genre_sum, index=['Other']))

# Colors: Customize the colors of the pie slices
colors = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99', '#c2c2c2', '#b3b3cc',␣
 ↪'#ffccff', '#ffb3e6',
          '#c2d6d6', '#d9d9d9', '#e6ffb3']
```

```
# Explode: Emphasize the top genres
explode = [0.1] * len(top_genres) + [0]

# Plotting the pie chart
plt.figure(figsize=(12, 8))
plt.pie(combined_genres, labels=combined_genres.index, explode=explode,␣
 ↪colors=colors,
        autopct='%.2f%%', textprops={'fontsize': 12})

plt.title('Worldwide Gross by Genre', fontsize=16)

# Creating a legend
plt.legend(title='Genres', loc='center left', bbox_to_anchor=(1, 0.5))

plt.tight_layout()

# Saving the figure with expanded bounding box
plt.savefig('pie_chart.png', bbox_inches='tight')

plt.show()
```

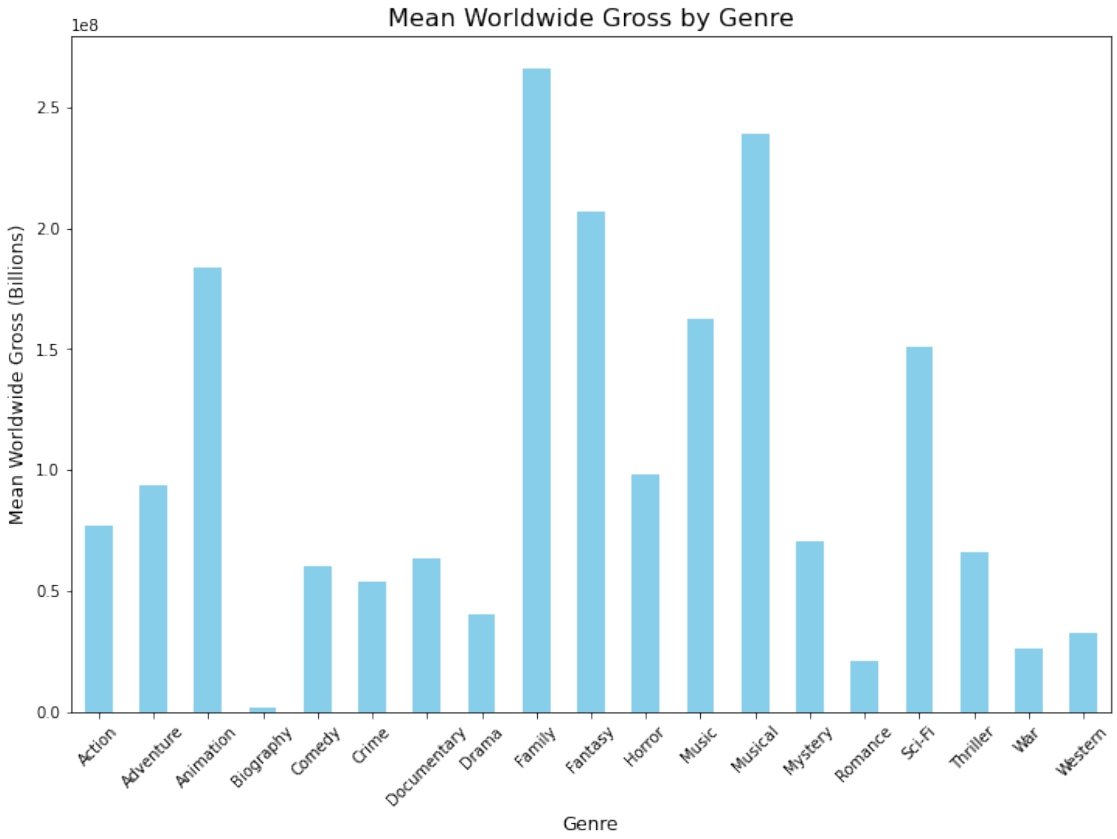


22 Summary statistics of Worldwide gross by Genre

Summary stats show that the mean gross return is highest for Sci-fi and Animation.

Summary stats show that Western and Biography return the less gross values.

[45]:
```
# Selected genres
selected_genres = ['News','Horror', 'Action', 'Adventure', 'Sci-Fi', 'Crime',␣
 ↪'Thriller', 'Drama', 'Fantasy', 'Romance', 'Animation', 'Comedy', 'Family',␣
 ↪'Musical', 'Biography', 'Music', 'Documentary', 'History', 'Mystery',␣
 ↪'Western', 'War', 'News']

# Subsetting the data for selected genres
subset_df = cleaned_df[cleaned_df['genres'].isin(selected_genres)]

# Grouping the data by genre and calculating summary statistics
genre_stats = subset_df.groupby('genres')['worldwide_gross'].describe()

# Plotting the mean worldwide gross for each genre
plt.figure(figsize=(12, 8))
genre_stats['mean'].plot(kind='bar', color='skyblue')
plt.title('Mean Worldwide Gross by Genre', fontsize=16)
plt.xlabel('Genre', fontsize=12)
plt.ylabel('Mean Worldwide Gross (Billions)', fontsize=12)
plt.xticks(rotation=45)
plt.show()

# Displaying the summary statistics
print(genre_stats)
```



genres	count	mean	std	min	25%	\
Action	16.0	7.718570e+07	8.883734e+07	0.0	8631032.75	
Adventure	7.0	9.387913e+07	1.854842e+08	0.0	2936282.50	
Animation	8.0	1.839455e+08	3.363916e+08	0.0	0.00	
Biography	5.0	1.906797e+06	3.212060e+06	0.0	32092.00	
Comedy	112.0	6.051115e+07	7.661146e+07	0.0	954560.50	
Crime	5.0	5.402849e+07	5.616716e+07	17639.0	4600000.00	
Documentary	246.0	6.350865e+07	1.235838e+08	0.0	1114772.25	
Drama	379.0	4.034965e+07	7.353359e+07	0.0	857725.00	
Family	12.0	2.661076e+08	6.579303e+08	0.0	48111.00	
Fantasy	4.0	2.065393e+08	2.485802e+08	1711.0	21010074.25	
Horror	73.0	9.845474e+07	3.431472e+08	0.0	478595.00	
Music	5.0	1.627513e+08	2.360596e+08	1381824.0	5149131.00	
Musical	7.0	2.389734e+08	2.286122e+08	531806.0	37998221.00	
Mystery	5.0	7.048935e+07	4.156773e+07	22673340.0	38253433.00	
Romance	13.0	2.122573e+07	2.110834e+07	0.0	4109095.00	
Sci-Fi	11.0	1.510624e+08	2.843647e+08	0.0	35867.50	
Thriller	56.0	6.630358e+07	1.430822e+08	0.0	1581316.25	
War	2.0	2.597455e+07	5.974419e+06	21750000.0	23862276.25	
Western	2.0	3.276629e+07	4.633853e+07	0.0	16383144.00	

```
                    50%           75%           max
genres
Action        52872764.0  1.075103e+08  3.424165e+08
Adventure      5953886.0  7.344462e+07  4.984382e+08
Animation     37767690.5  1.468067e+08  9.628545e+08
Biography       719699.0  1.200000e+06  7.582196e+06
Comedy        28139824.0  9.576641e+07  3.360695e+08
Crime         41076865.0  9.962487e+07  1.248231e+08
Documentary   15416508.0  7.293425e+07  1.025491e+09
Drama         11295324.0  4.726783e+07  4.875198e+08
Family         5423953.5  3.928192e+07  2.208208e+09
Fantasy      145802138.5  3.313314e+08  5.345514e+08
Horror        16340767.0  6.524551e+07  2.776345e+09
Music          9082906.0  2.635914e+08  5.345514e+08
Musical      263591415.0  3.990714e+08  5.345514e+08
Mystery       81079566.0  8.218368e+07  1.282567e+08
Romance       19535005.0  2.670318e+07  5.897848e+07
Sci-Fi        28876702.0  7.559126e+07  8.211334e+08
Thriller      18805528.5  6.932313e+07  8.536286e+08
War           25974552.5  2.808683e+07  3.019910e+07
Western       32766288.0  4.914943e+07  6.553258e+07
```

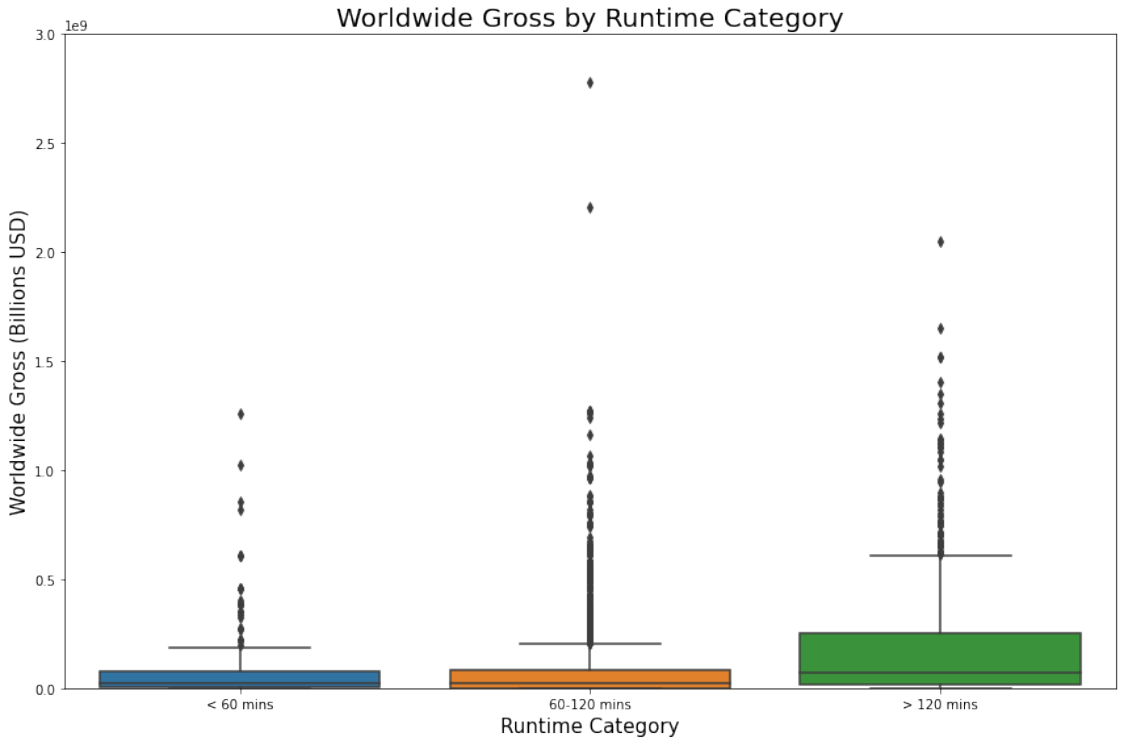
23 Reviewing runtimes effect on Worldwide gross

Seperated the runtime data into 3 subcatagories short(<60 mins), med (60-120 mins) , long (120+ mins)

Long movies tend to return the highest worldwide gross. Meduium movies and short return lowest worldwide gross.

[46]:
```
# Define the runtime categories and their corresponding labels
runtime_bins = [0, 60, 120, float('inf')]
runtime_labels = ['< 60 mins', '60-120 mins', '> 120 mins']

# Create the runtime_category column
cleaned_df['runtime_category'] = pd.cut(cleaned_df['runtime_minutes'],␣
 ↪bins=runtime_bins, labels=runtime_labels)

# Plotting worldwide gross by runtime category
plt.figure(figsize=(12, 8))
sns.boxplot(x=cleaned_df['runtime_category'], y=cleaned_df['worldwide_gross'])
plt.title('Worldwide Gross by Runtime Category', fontsize=20)
plt.xlabel('Runtime Category', fontsize=15)
plt.ylabel('Worldwide Gross (Billions USD)', fontsize=15)

# Set y-axis limits
plt.ylim(0, 3e9)  # Set y-axis limit to 3 billion
```

```
# Set y-axis ticks at desired intervals
plt.yticks([0, 5e8, 1e9, 1.5e9, 2e9, 2.5e9, 3e9])

plt.tight_layout()  # Adjust spacing between subplots

plt.show()
```



24 Summary statistics for runtimes impact on Worldwide gross

The chart shows that the summary stats are consistent with the box plot. Long movies return higher gross, while short and medium return less gross.

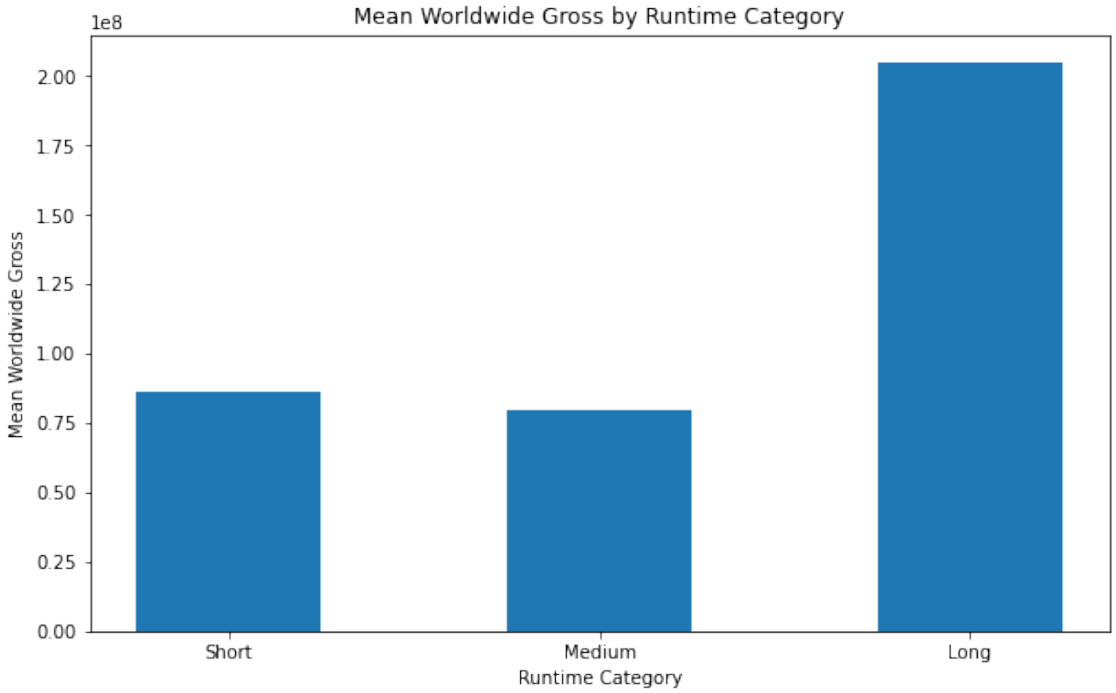
[47]:
```
# Calculate summary statistics for ww gross by runtime category
runtime_summary = cleaned_df.groupby('runtime_category')['worldwide_gross'].
  ↪describe()

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

```
                   count          mean           std  min         25%  \
runtime_category
< 60 mins          282.0  8.620738e+07  1.622639e+08  0.0   2788004.5
```

```
60-120 mins       2549.0  7.961783e+07  1.636566e+08  0.0   1200000.0
> 120 mins         478.0  2.045094e+08  3.099314e+08  0.0  16446364.0

                         50%           75%           max
runtime_category
< 60 mins          25653962.5  8.016036e+07  1.259200e+09
60-120 mins        22673340.0  8.332000e+07  2.776345e+09
> 120 mins         72912745.5  2.547346e+08  2.048134e+09
```

25 Summary stats for runtimes impact on Worldwide Gross

[48]:
```
# Group the data by runtime category and calculate the mean worldwide gross
mean_values = cleaned_df.groupby('runtime_category')['worldwide_gross'].mean()

# Define the categories
categories = ['Short', 'Medium', 'Long']

# Set the width of the bars
bar_width = 0.5

# Plotting the mean values for each category
plt.figure(figsize=(10, 6))
plt.bar(categories, mean_values, width=bar_width)

plt.title('Mean Worldwide Gross by Runtime Category')
plt.xlabel('Runtime Category')
plt.ylabel('Mean Worldwide Gross')
plt.show()
```



[49]:
```
print(cleaned_df.columns)
```

```
Index(['movie', 'production_budget', 'worldwide_gross', 'runtime_minutes',
       'genres', 'runtime_category'],
      dtype='object')
```

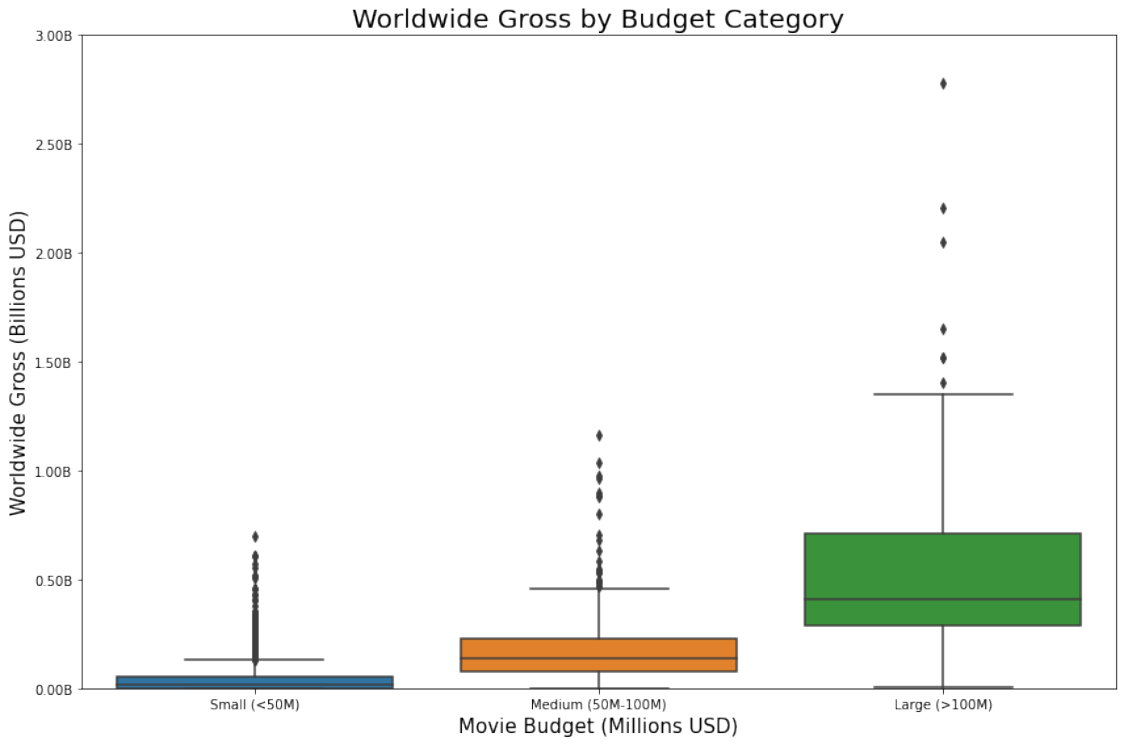
26 Production Budgets impact on Worldwide Gross

Broke up movie budgets into three catagories Small (< 50 Million), Med (50-100 Million), and Large (>100 Million)

Large budgets generated the most worldwide gross, including movies that generated over 2 billion USD. Small budgets did not generate a single movie over 1 billion dollars. Medium budgets did better than small budgets and generated a few movies over a billion dollars.

[50]:
```
# Y axis Billions
def billions_formatter(x, pos):
    return f'{x/1e9:.2f}B'

# Categorize the movie budgets into small, medium, and large
budget_labels = ['Small (<50M)', 'Medium (50M-100M)', 'Large (>100M)']
bins = [0, 50e6, 100e6, float('inf')]
cleaned_df['budget_category'] = pd.cut(cleaned_df['production_budget'],␣
 ↪bins=bins, labels=budget_labels)
```

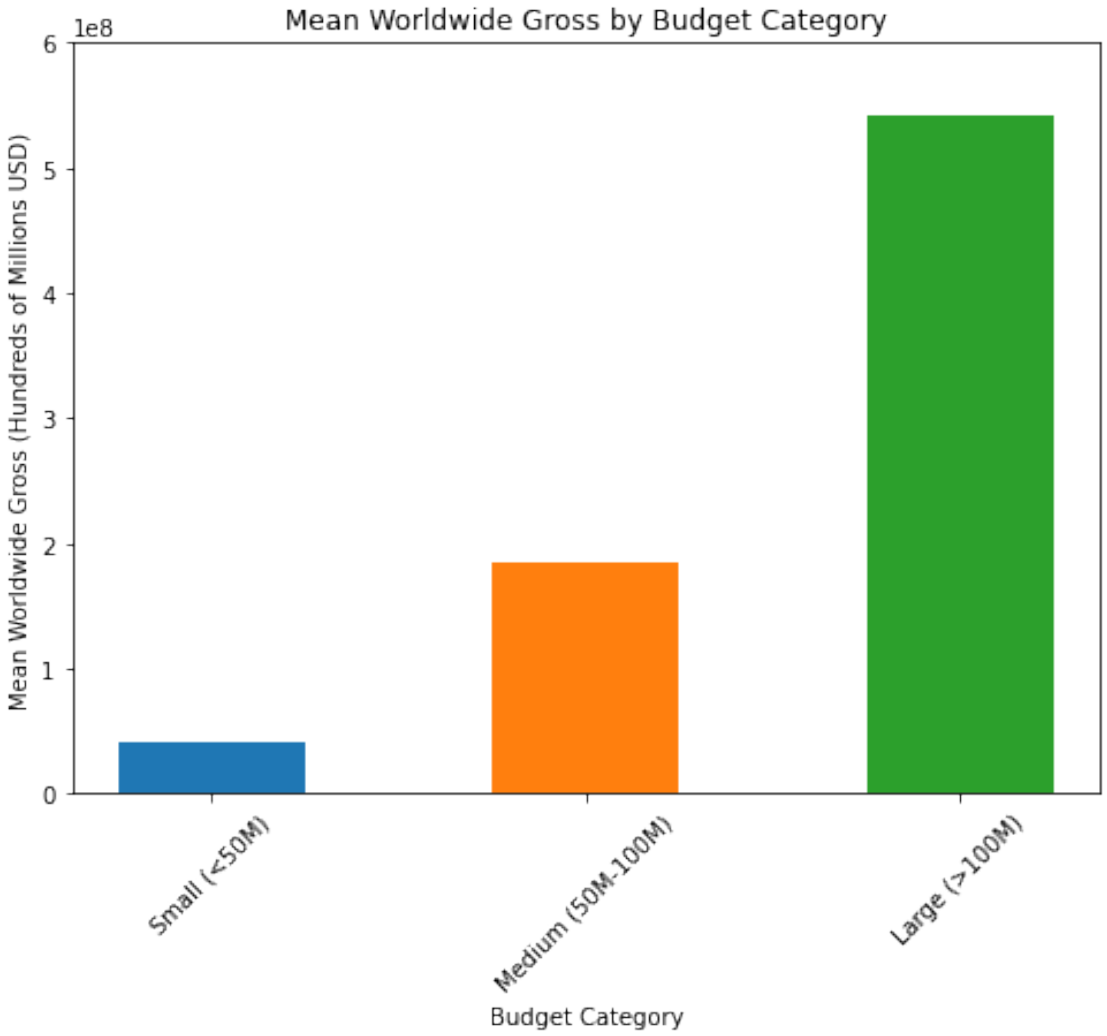
```
# Plotting worldwide gross by budget category
plt.figure(figsize=(12, 8))
sns.boxplot(x=cleaned_df['budget_category'], y=cleaned_df['worldwide_gross'])
plt.title('Worldwide Gross by Budget Category', fontsize=20)
plt.xlabel('Movie Budget (Millions USD)', fontsize=15)
plt.ylabel('Worldwide Gross (Billions USD)', fontsize=15)

# Set y-axis limits-#3b
plt.ylim(0, 3e9)

# Format y-axis labels as billions
formatter = FuncFormatter(billions_formatter)
ax = plt.gca()
ax.yaxis.set_major_formatter(formatter)

plt.ylim(0, 3e9)  # Set y-axis limit to 3 billion


plt.tight_layout()  # Adjust spacing between subplots

plt.show()
```



27 Average worldwide gross by budget

Large budgets produced on average about 550,000,000 Med budgets produced on average about 180,000,000 Small budget worldwide gross. 40,000,000

[51]:
```
# Group the data by budget category and calculate the mean worldwide gross
mean_values = cleaned_df.groupby('budget_category')['worldwide_gross'].mean()

# Define the categories
categories = ['Small (<50M)', 'Medium (50M-100M)', 'Large (>100M)']

# Set the width of the bars
bar_width = 0.5

# Plotting the mean values for each category
plt.figure(figsize=(8, 6))
plt.bar(categories, mean_values, width=bar_width, color=['#1f77b4', '#ff7f0e',␣
 ↪'#2ca02c'])

plt.title('Mean Worldwide Gross by Budget Category')
plt.xlabel('Budget Category')
plt.ylabel('Mean Worldwide Gross (Millions USD)')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability

plt.ylim(0, 600e6)  # Set y-axis limit to 800 million
plt.ylabel('Mean Worldwide Gross (Hundreds of Millions USD)')




plt.show()
```



[52]:
```
# Calculate summary statistics for ww gross by budget category
budget_summary = cleaned_df.groupby('budget_category')['worldwide_gross'].
 ↪describe()

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

```
                    count          mean           std        min          25%  \
budget_category
Small (<50M)       2676.0  4.118832e+07  6.860501e+07        0.0     740932.0
Medium (50M-100M)   359.0  1.853351e+08  1.753544e+08        0.0   77592282.5
Large (>100M)       274.0  5.410824e+08  3.780591e+08  3100000.0  290048571.0

                           50%           75%           max
budget_category
```

```
Small (<50M)         14868357.5  5.246054e+07  6.974580e+08
Medium (50M-100M)  137489730.0  2.303680e+08  1.160336e+09
Large (>100M)      409378800.5  7.134626e+08  2.776345e+09
```

28 Conclusion

28.1 Genre's impact on Worldwide gross

Sci-Fi, Animation, Adventure, action (based on box-plot) produces the largest Worldwide Gross Genre such as western, bio, musical, romance, news are low Worldwide Gross earners.

28.2 Runtime's impact on Worldwide Gross

Long runtimes are better for gross. Short and med do not produce high Worldwide gross.

28.3 Budgets impact on Worldwide Gross

Higher budget increases likelihood of higher Worldwide Gross. Small budget does not produce billion dollar gross movies. A few medium budget movies have produced over a billion dollars in Worldwide Gross.

28.4 Recommendations

The movies genre should be a mixture of Sci-Fi, Animation, Adventure and action. Microsoft Studios should not make Westerns, biographies, musicals or new films. The runtime of the movies should be over two hours long. A budget larger than 100 million dollars will produce the highest Worldwide Gross.

[ ]: