

# #Importing Relevant Libraries

```
In [1]: import numpy as np
import pandas as pd
import sqlite3
# importing the relevant data visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns
from glob import glob
```

# #Loading Data

```
In [2]: csv_files = glob("./zippedData/*.csv.gz")
csv_files
```

```
Out[2]: ['./zippedData\bom.movie_gross.csv.gz',
 './zippedData\tmdb.movies.csv.gz',
 './zippedData\tn.movie_budgets.csv.gz']
```

```
In [3]: bom_df=pd.read_csv('./zippedData\bom.movie_gross.csv.gz')
```

```
In [4]: budgets_df=pd.read_csv('./zippedData\tn.movie_budgets.csv.gz')
```

# #Data Understanding(budgets)

```
In [5]: budgets_df.head()
```

	<b>id</b>	<b>release_date</b>	<b>movie</b>	<b>production_budget</b>	<b>domestic_gross</b>	<b>worldwide_gross</b>
<b>0</b>	1	Dec 18, 2009	Avatar	\$425,000,000	\$760,507,625	\$2,776,345,279
<b>1</b>	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	\$410,600,000	\$241,063,875	\$1,045,663,875
<b>2</b>	3	Jun 7, 2019	Dark Phoenix	\$350,000,000	\$42,762,350	\$149,762,350
<b>3</b>	4	May 1, 2015	Avengers: Age of Ultron	\$330,600,000	\$459,005,868	\$1,403,013,963
<b>4</b>	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	\$317,000,000	\$620,181,382	\$1,316,721,747

```
In [6]: budgets_df.tail()
```

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

In [7]: `budgets_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 [8]: `budgets_df.describe()`

Out[8]: **id**

	<b>count</b>
	5782.000000
	<b>mean</b>
	50.372363
	<b>std</b>
	28.821076
	<b>min</b>
	1.000000
	<b>25%</b>
	25.000000
	<b>50%</b>
	50.000000
	<b>75%</b>
	75.000000
	<b>max</b>
	100.000000

There are no null values.

## #Cleaning(budgets)

In [9]: `#checking for duplicates  
budgets_df.duplicated().any()`

Out[9]: `False`

In [10]: `budgets_df['production_budget'][1]`

Out[10]: `'$410,600,000'`

In [11]: `#converting to a numeric data type  
def numclean(df,col):  
 df[col]=df[col].str.replace("$","",).str.replace(",","",).astype(float)  
 return df  
numclean(budgets_df,'production_budget')  
numclean(budgets_df,'domestic_gross')  
numclean(budgets_df,'worldwide_gross')`

Out[11]:

	<b>id</b>	<b>release_date</b>	<b>movie</b>	<b>production_budget</b>	<b>domestic_gross</b>	<b>worldwide_gross</b>
<b>0</b>	1	Dec 18, 2009	Avatar	425000000.0	760507625.0	2.776345e+09
<b>1</b>	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09

	<b>id</b>	<b>release_date</b>	<b>movie</b>	<b>production_budget</b>	<b>domestic_gross</b>	<b>worldwide_gross</b>
2	3	Jun 7, 2019	Dark Phoenix	3500000000.0	42762350.0	1.497624e+08
3	4	May 1, 2015	Avengers: Age of Ultron	3306000000.0	459005868.0	1.403014e+09
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	3170000000.0	620181382.0	1.316722e+09
...	...	...	...	...	...	...
5777	78	Dec 31, 2018	Red 11	7000.0	0.0	0.000000e+00
5778	79	Apr 2, 1999	Following	6000.0	48482.0	2.404950e+05
5779	80	Jul 13, 2005	Return to the Land of Wonders	5000.0	1338.0	1.338000e+03
5780	81	Sep 29, 2015	A Plague So Pleasant	1400.0	0.0	0.000000e+00
5781	82	Aug 5, 2005	My Date With Drew	1100.0	181041.0	1.810410e+05

5782 rows × 6 columns

In [12]: budgets\_df.head()

	<b>id</b>	<b>release_date</b>	<b>movie</b>	<b>production_budget</b>	<b>domestic_gross</b>	<b>worldwide_gross</b>
0	1	Dec 18, 2009	Avatar	425000000.0	760507625.0	2.776345e+09
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09
2	3	Jun 7, 2019	Dark Phoenix	350000000.0	42762350.0	1.497624e+08
3	4	May 1, 2015	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000.0	620181382.0	1.316722e+09

In [13]: budgets\_df.tail()

	<b>id</b>	<b>release_date</b>	<b>movie</b>	<b>production_budget</b>	<b>domestic_gross</b>	<b>worldwide_gross</b>
5777	78	Dec 31, 2018	Red 11	7000.0	0.0	0.0
5778	79	Apr 2, 1999	Following	6000.0	48482.0	240495.0
5779	80	Jul 13, 2005	Return to the Land of Wonders	5000.0	1338.0	1338.0
5780	81	Sep 29, 2015	A Plague So Pleasant	1400.0	0.0	0.0
5781	82	Aug 5, 2005	My Date With Drew	1100.0	181041.0	181041.0

In [14]: budgets\_cleaned=budgets\_df.drop('id',axis=1)  
budgets\_cleaned.head()

	<b>release_date</b>	<b>movie</b>	<b>production_budget</b>	<b>domestic_gross</b>	<b>worldwide_gross</b>
0	Dec 18, 2009	Avatar	425000000.0	760507625.0	2.776345e+09

	release_date	movie	production_budget	domestic_gross	worldwide_gross
1	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09
2	Jun 7, 2019	Dark Phoenix	350000000.0	42762350.0	1.497624e+08
3	May 1, 2015	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09
4	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000.0	620181382.0	1.316722e+09

```
In [15]: #convert release date to date_time format
budgets_cleaned['release_date'] = pd.to_datetime(budgets_cleaned['release_date'])
budgets_cleaned.head()
```

	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	2009-12-18	Avatar	425000000.0	760507625.0	2.776345e+09
1	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09
2	2019-06-07	Dark Phoenix	350000000.0	42762350.0	1.497624e+08
3	2015-05-01	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09
4	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000.0	620181382.0	1.316722e+09

## #Data Understanding(Box office)

```
In [16]: bom_df.head()
```

	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

```
In [17]: bom_df.tail()
```

	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 [18]: bom_df.shape
```

(3387, 5)

Out[18]:

In [19]: `bom_df.describe()`

	<b>domestic_gross</b>	<b>year</b>
<b>count</b>	3.359000e+03	3387.000000
<b>mean</b>	2.874585e+07	2013.958075
<b>std</b>	6.698250e+07	2.478141
<b>min</b>	1.000000e+02	2010.000000
<b>25%</b>	1.200000e+05	2012.000000
<b>50%</b>	1.400000e+06	2014.000000
<b>75%</b>	2.790000e+07	2016.000000
<b>max</b>	9.367000e+08	2018.000000

In [20]: `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
```

There are null values in the following columns: studio  
 domestic\_gross foreign\_gross

## #Data Cleaning(Box Office)

In [21]: `# number of rows with null values  
bom_df.isnull().sum()`

```
Out[21]: title      0
studio     5
domestic_gross 28
foreign_gross 1350
year       0
dtype: int64
```

In [22]: `# sample rows where foreign_gross has no missing values  
bom_df[bom_df["studio"].notna()].sample(5, random_state = 1)`

	<b>title</b>	<b>studio</b>	<b>domestic_gross</b>	<b>foreign_gross</b>	<b>year</b>
<b>571</b>	Poetry	Kino	356000.0	1900000	2011
<b>3343</b>	The Third Murder	FM	89300.0	NaN	2018
<b>1850</b>	Camp X-Ray	IFC	13300.0	NaN	2014
<b>3257</b>	Don't Worry He Won't Get Far on Foot	Amazon	1400000.0	2500000	2018

	title	studio	domestic_gross	foreign_gross	year
1512	Non-Stop	Uni.	92200000.0	130600000	2014

In [23]: `# sample rows where foreign_gross has missing values  
bom_df[bom_df["studio"].isna()]`

	title	studio	domestic_gross	foreign_gross	year
210	Outside the Law (Hors-la-loi)	NaN	96900.0	3300000	2010
555	Fireflies in the Garden	NaN	70600.0	3300000	2011
933	Keith Lemon: The Film	NaN		4000000	2012
1862	Plot for Peace	NaN	7100.0		2014
2825	Secret Superstar	NaN		122000000	2017

In [24]: `bom_df.dropna(subset = ["studio"], inplace = True)`

In [25]: `bom_df.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3382 entries, 0 to 3386
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   title            3382 non-null   object 
 1   studio           3382 non-null   object 
 2   domestic_gross   3356 non-null   float64
 3   foreign_gross   2033 non-null   object 
 4   year             3382 non-null   int64  
dtypes: float64(1), int64(1), object(3)
memory usage: 158.5+ KB
```

In [26]: `# percentage of missing values in domestic_gross  
bom_df["domestic_gross"].isnull().sum()/len(bom_df)*100`

Out[26]: 0.768775872264932

Since the percentage is quite small, we can drop the rows.

In [27]: `bom_df.dropna(subset = ["domestic_gross"], inplace = True)  
bom_df.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   2007 non-null   object 
 4   year             3356 non-null   int64  
dtypes: float64(1), int64(1), object(3)
memory usage: 157.3+ KB
```

In [28]: `# percentage of missing values in foreign gross  
bom_df["foreign_gross"].isnull().sum()/len(bom_df)*100`

Out[28]: 40.19666269368295

Not droppable as it is too large. We shall replace the values with either the mean or median depending on our investigation.

```
In [29]: # preview of foreign gross column
bom_df["foreign_gross"].head()
```

```
Out[29]: 0    652000000
1    691300000
2    664300000
3    535700000
4    513900000
Name: foreign_gross, dtype: object
```

```
In [30]: #converting to numerical values
bom_df["foreign_gross"] = [float(str(i).replace(",", "")) for i in bom_df["foreign_gross"]]
bom_df.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    2007 non-null   float64 
 4   year             3356 non-null   int64  
dtypes: float64(2), int64(1), object(2)
memory usage: 157.3+ KB
```

```
In [31]: bom_df["foreign_gross"].mean()
```

```
Out[31]: 75790384.84130543
```

```
In [32]: bom_df["foreign_gross"].median()
```

```
Out[32]: 19400000.0
```

```
In [33]: #replacing with mean as replacing with the median affects mean
bom_df["foreign_gross"].fillna(bom_df["foreign_gross"].mean(), inplace=True)
bom_df["foreign_gross"].mean()
```

```
Out[33]: 75790384.84130543
```

```
In [34]: bom_df.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
```

```
In [35]: bom_df.duplicated().any()
```

```
Out[35]: False
```

```
In [36]: bom_df.describe()
```

Out[36]:

	domestic_gross	foreign_gross	year
<b>count</b>	3.356000e+03	3.356000e+03	3356.000000
<b>mean</b>	2.877149e+07	7.579038e+07	2013.970203
<b>std</b>	6.700694e+07	1.068472e+08	2.479064
<b>min</b>	1.000000e+02	6.000000e+02	2010.000000
<b>25%</b>	1.200000e+05	1.220000e+07	2012.000000
<b>50%</b>	1.400000e+06	7.579038e+07	2014.000000
<b>75%</b>	2.795000e+07	7.579038e+07	2016.000000
<b>max</b>	9.367000e+08	9.605000e+08	2018.000000

## #Data Understanding(imdb)

In [37]:

```
conn= sqlite3.connect("./zippedData/im.db_1/im.db")
```

In [38]:

```
cur=conn.cursor()
```

In [39]:

```
cur.execute("""SELECT name FROM sqlite_master WHERE type = 'table';""")
table_names = cur.fetchall()
table_names
```

Out[39]:

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

In [40]:

```
moviebasics_df=pd.read_sql("""
SELECT * FROM movie_basics;
""", conn)
```

In [41]:

```
movieratings_df=pd.read_sql("""
SELECT * FROM movie_ratings;
""", conn)
```

## #Data Cleaning(movie\_basics)

In [42]:

```
pd.read_sql("""
SELECT * FROM movie_basics;
""", conn).head().head()
```

Out[42]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres
<b>0</b>	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
<b>1</b>	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama
<b>2</b>	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama

	<b>movie_id</b>	<b>primary_title</b>	<b>original_title</b>	<b>start_year</b>	<b>runtime_minutes</b>	<b>genres</b>
<b>3</b>	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
<b>4</b>	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy

```
In [43]: pd.read_sql("""
SELECT * FROM movie_basics;
""", conn).head().tail()
```

	<b>movie_id</b>	<b>primary_title</b>	<b>original_title</b>	<b>start_year</b>	<b>runtime_minutes</b>	<b>genres</b>
<b>0</b>	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
<b>1</b>	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama
<b>2</b>	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama
<b>3</b>	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
<b>4</b>	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy

```
In [44]: pd.read_sql("""
SELECT * FROM movie_basics;
""", conn).info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   movie_id         146144 non-null   object 
 1   primary_title    146144 non-null   object 
 2   original_title   146123 non-null   object 
 3   start_year       146144 non-null   int64  
 4   runtime_minutes  114405 non-null   float64
 5   genres           140736 non-null   object 
dtypes: float64(1), int64(1), object(4)
memory usage: 6.7+ MB
```

Discrepancies in entries indicate existence of null values.

```
In [45]: pd.read_sql("""
SELECT * FROM movie_basics;
""", conn).describe()
```

	<b>start_year</b>	<b>runtime_minutes</b>
<b>count</b>	146144.000000	114405.000000
<b>mean</b>	2014.621798	86.187247
<b>std</b>	2.733583	166.360590
<b>min</b>	2010.000000	1.000000
<b>25%</b>	2012.000000	70.000000
<b>50%</b>	2015.000000	87.000000

	start_year	runtime_minutes
75%	2017.000000	99.000000
max	2115.000000	51420.000000

In [46]: `moviebasics_df.isna().sum()`

Out[46]:

movie_id	0
primary_title	0
original_title	21
start_year	0
runtime_minutes	31739
genres	5408
dtype:	int64

In [47]: `((moviebasics_df["original_title"].isna().sum()/len(moviebasics_df)) * 100)`

Out[47]: 0.014369389095686446

In [48]: *#Drop as percentage is insignificant*  
`moviebasics_df.dropna(subset = ["original_title"], inplace = True)`

In [49]: `moviebasics_df.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 146123 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   movie_id         146123 non-null   object 
 1   primary_title    146123 non-null   object 
 2   original_title   146123 non-null   object 
 3   start_year       146123 non-null   int64  
 4   runtime_minutes  114401 non-null   float64
 5   genres           140734 non-null   object 
dtypes: float64(1), int64(1), object(4)
memory usage: 7.8+ MB
```

In [50]: `((moviebasics_df["runtime_minutes"].isna().sum()/len(moviebasics_df)) * 100)`

Out[50]: 21.709108080179025

replacing with either mean or median as it is a substantial percentage

In [51]: `moviebasics_df["runtime_minutes"].mean()`

Out[51]: 86.18612599540214

In [52]: `moviebasics_df["runtime_minutes"].median()`

Out[52]: 87.0

In [53]: `moviebasics_df["runtime_minutes"].fillna(moviebasics_df["runtime_minutes"].mean(),in`

In [54]: *#investigating effect on mean*  
`moviebasics_df["runtime_minutes"].mean()`

Out[54]: 86.18612599540215

In [55]: `moviebasics_df.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 146123 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype  
--- 
 0   movie_id         146123 non-null    object  
 1   primary_title    146123 non-null    object  
 2   original_title   146123 non-null    object  
 3   start_year       146123 non-null    int64   
 4   runtime_minutes  146123 non-null    float64 
 5   genres           140734 non-null    object  
dtypes: float64(1), int64(1), object(4)
memory usage: 7.8+ MB
```

In [56]: `((moviebasics_df["genres"].isna().sum())/len(moviebasics_df)) * 100)`

Out[56]: 3.6879888860754293

In [57]: `#Dropping as value is insignificant  
moviebasics_df.dropna(subset = ["genres"], inplace = True)  
moviebasics_df.info()`

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 140734 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype  
--- 
 0   movie_id         140734 non-null    object  
 1   primary_title    140734 non-null    object  
 2   original_title   140734 non-null    object  
 3   start_year       140734 non-null    int64   
 4   runtime_minutes  140734 non-null    float64 
 5   genres           140734 non-null    object  
dtypes: float64(1), int64(1), object(4)
memory usage: 7.5+ MB
```

In [58]: `moviebasics_df.isna().sum()`

Out[58]: `movie_id 0  
primary_title 0  
original_title 0  
start_year 0  
runtime_minutes 0  
genres 0  
dtype: int64`

In [59]: `moviebasics_df.duplicated().any()`

Out[59]: False

## #Data Cleaning(movie\_ratings)

In [60]: `movieratings_df.head()`

	movie_id	averagerating	numvotes
<b>0</b>	tt10356526	8.3	31
<b>1</b>	tt10384606	8.9	559
<b>2</b>	tt1042974	6.4	20
<b>3</b>	tt1043726	4.2	50352
<b>4</b>	tt1060240	6.5	21

In [61]: `movieratings_df.tail()`

Out[61]:

	<b>movie_id</b>	<b>averagerating</b>	<b>numvotes</b>
<b>73851</b>	tt9805820	8.1	25
<b>73852</b>	tt9844256	7.5	24
<b>73853</b>	tt9851050	4.7	14
<b>73854</b>	tt9886934	7.0	5
<b>73855</b>	tt9894098	6.3	128

In [62]: `movieratings_df.shape`

Out[62]: (73856, 3)

In [63]: `movieratings_df.describe()`

Out[63]:

	<b>averagerating</b>	<b>numvotes</b>
<b>count</b>	73856.000000	7.385600e+04
<b>mean</b>	6.332729	3.523662e+03
<b>std</b>	1.474978	3.029402e+04
<b>min</b>	1.000000	5.000000e+00
<b>25%</b>	5.500000	1.400000e+01
<b>50%</b>	6.500000	4.900000e+01
<b>75%</b>	7.400000	2.820000e+02
<b>max</b>	10.000000	1.841066e+06

In [64]: `movieratings_df.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73856 entries, 0 to 73855
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype  
 --- 
 0   movie_id    73856 non-null   object 
 1   averagerating 73856 non-null   float64
 2   numvotes     73856 non-null   int64  
dtypes: float64(1), int64(1), object(1)
memory usage: 1.7+ MB
```

There are no null values.

In [65]: `movieratings_df.duplicated().any()`

Out[65]: False

## #DATA ANALYSIS

### Budgets

In [66]: `budgets_df.head()`

	<b>id</b>	<b>release_date</b>	<b>movie</b>	<b>production_budget</b>	<b>domestic_gross</b>	<b>worldwide_gross</b>
<b>0</b>	1	Dec 18, 2009	Avatar	425000000.0	760507625.0	2.776345e+09
<b>1</b>	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09
<b>2</b>	3	Jun 7, 2019	Dark Phoenix	350000000.0	42762350.0	1.497624e+08
<b>3</b>	4	May 1, 2015	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09
<b>4</b>	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000.0	620181382.0	1.316722e+09

In [67]: `# Plot release day for all movies  
budgets_df['production_budget'].describe()`

Out[67]:

count	5.782000e+03
mean	3.158776e+07
std	4.181208e+07
min	1.100000e+03
25%	5.000000e+06
50%	1.700000e+07
75%	4.000000e+07
max	4.250000e+08

Name: production\_budget, dtype: float64

In [68]: `budgets_df[['movie', 'production_budget']]`

	<b>movie</b>	<b>production_budget</b>
<b>0</b>	Avatar	425000000.0
<b>1</b>	Pirates of the Caribbean: On Stranger Tides	410600000.0
<b>2</b>	Dark Phoenix	350000000.0
<b>3</b>	Avengers: Age of Ultron	330600000.0
<b>4</b>	Star Wars Ep. VIII: The Last Jedi	317000000.0
...	...	...
<b>5777</b>	Red 11	7000.0
<b>5778</b>	Following	6000.0
<b>5779</b>	Return to the Land of Wonders	5000.0
<b>5780</b>	A Plague So Pleasant	1400.0
<b>5781</b>	My Date With Drew	1100.0

5782 rows × 2 columns

In [69]: `budgets_df['production_budget'].describe().apply(lambda x: format(x, 'f'))`

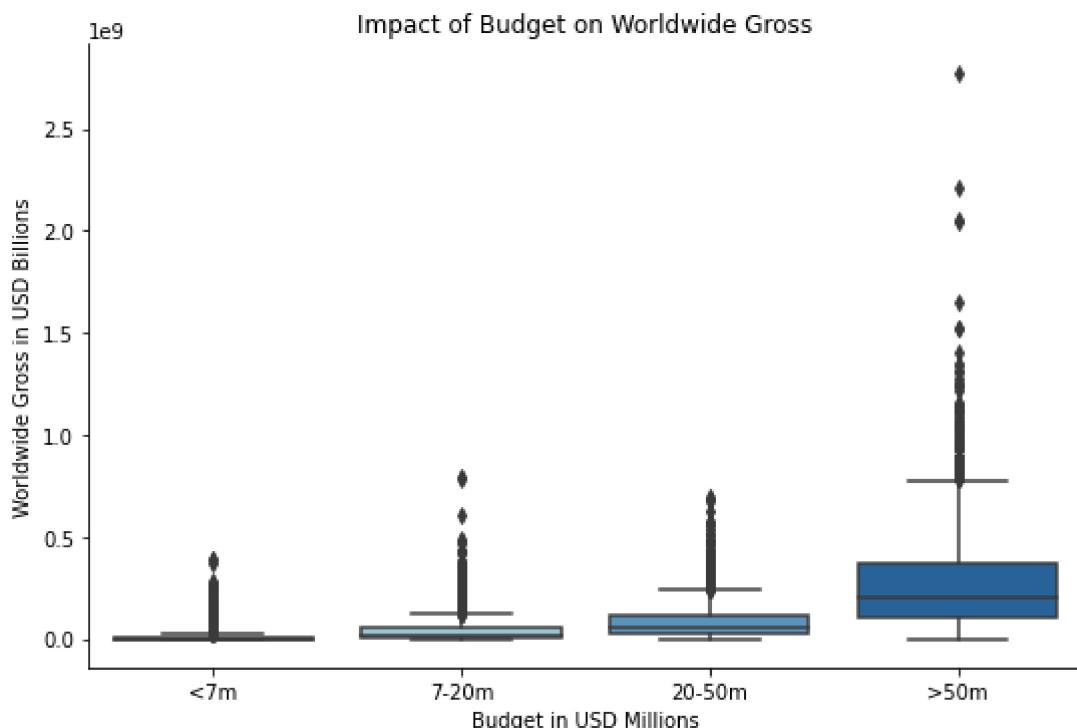
Out[69]:

count	5782.000000
mean	31587757.096506
std	41812076.826943
min	1100.000000
25%	5000000.000000
50%	17000000.000000
75%	40000000.000000

```
max      425000000.00000
Name: production_budget, dtype: object
```

```
In [70]: bins = [0, 7000000, 20000000, 50000000, np.inf]
names = ['<7m', '7-20m', '20-50m', '>50m']
budgets_df['budget_range'] = pd.cut(budgets_df['production_budget'], bins, labels=na
```

```
In [71]: #plotting to see the budget range
sns.catplot(x = 'budget_range', y = 'worldwide_gross', aspect = 1.5, kind = 'box',
            palette="Blues", data = budgets_df)
plt.title('Impact of Budget on Worldwide Gross')
plt.xlabel('Budget in USD Millions')
plt.ylabel('Worldwide Gross in USD Billions')
plt.show()
```



## #Recommendation based on budgets

A budget of above \$50 million leads to a greater worldwide gross thus higher profits

## #Release Month

```
In [72]: # Create month column
budgets_df['release_month'] = pd.DatetimeIndex(budgets_df['release_date']).month
```

```
In [73]: budgets_df.head()
```

	<b>id</b>	<b>release_date</b>	<b>movie</b>	<b>production_budget</b>	<b>domestic_gross</b>	<b>worldwide_gross</b>	<b>budget_range</b>
<b>0</b>	1	Dec 18, 2009	Avatar	425000000.0	760507625.0	2.776345e+09	>50m
<b>1</b>	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09	>50m

	<b>id</b>	<b>release_date</b>	<b>movie</b>	<b>production_budget</b>	<b>domestic_gross</b>	<b>worldwide_gross</b>	<b>budget_range</b>
<b>2</b>	3	Jun 7, 2019	Dark Phoenix	3500000000.0	42762350.0	1.497624e+08	>50m
<b>3</b>	4	May 1, 2015	Avengers: Age of Ultron	3306000000.0	459005868.0	1.403014e+09	>50m
<b>4</b>	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	3170000000.0	620181382.0	1.316722e+09	>50m

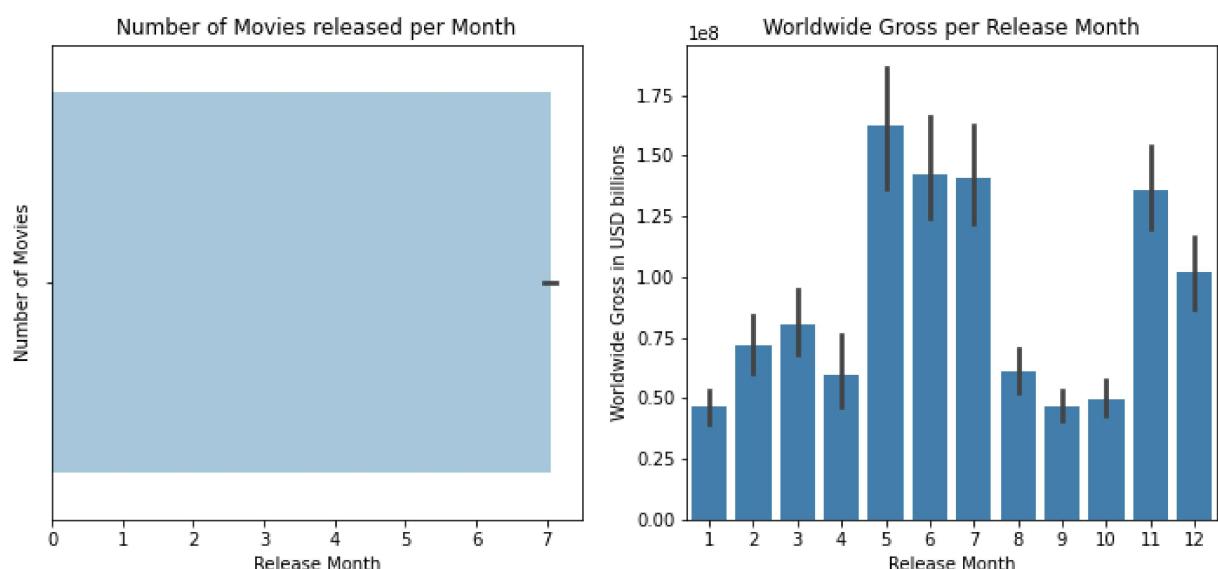


In [74]:

```
def releasemonthplot(df):
    fig, ax = plt.subplots(ncols = 2, nrows = 1, figsize = (12,5))
    sns.barplot( x = 'release_month', color = '#9ecae1',data = df, ax = ax[0])
    ax[0].set_xlabel('Release Month')
    ax[0].set_ylabel('Number of Movies')
    ax[0].set_title('Number of Movies released per Month')
    sns.barplot( x = 'release_month', y = 'worldwide_gross', color = '#3182bd',
                data = df, ax = ax[1])
    ax[1].set_xlabel('Release Month')
    ax[1].set_ylabel('Worldwide Gross in USD billions')
    ax[1].set_title('Worldwide Gross per Release Month')
    plt.close(2)
    plt.close(3)
    return plt.show()
```

In [75]:

```
#Running function for dataframe
releasemonthplot(budgets_df)
```



## #Recommendation(Release month)

There is generally a higher worldwide gross in June and July thus one should aim to release movies then

## #Data Analysis(Runtime)

In [76]:

```
moviebasics_df.head()
```

Out[76]:

	<b>movie_id</b>	<b>primary_title</b>	<b>original_title</b>	<b>start_year</b>	<b>runtime_minutes</b>	<b>genres</b>
<b>0</b>	tt0063540	Sunghursh	Sunghursh	2013	175.000000	Action,Crime,Drama
<b>1</b>	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.000000	Biography,Drama
<b>2</b>	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.000000	Drama
<b>3</b>	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	86.186126	Comedy,Drama
<b>4</b>	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.000000	Comedy,Drama,Fantasy

In [77]: `moviebasics_df[['primary_title', 'runtime_minutes']]`

Out[77]:

	<b>primary_title</b>	<b>runtime_minutes</b>
<b>0</b>	Sunghursh	175.000000
<b>1</b>	One Day Before the Rainy Season	114.000000
<b>2</b>	The Other Side of the Wind	122.000000
<b>3</b>	Sabse Bada Sukh	86.186126
<b>4</b>	The Wandering Soap Opera	80.000000
...	...	...
<b>146138</b>	The Secret of China	86.186126
<b>146139</b>	Kuambil Lagi Hatiku	123.000000
<b>146140</b>	Rodolpho Teóphilo - O Legado de um Pioneiro	86.186126
<b>146141</b>	Dankyavar Danka	86.186126
<b>146143</b>	Chico Albuquerque - Revelações	86.186126

140734 rows × 2 columns

In [78]: `moviebasics_df['runtime_minutes'].describe()`

Out[78]:

count	140734.000000
mean	86.246280
std	149.934119
min	1.000000
25%	75.000000
50%	86.186126
75%	95.000000
max	51420.000000
Name:	runtime_minutes, dtype: float64

In [79]: `moviebasics_df.rename(columns = {'primary_title':'movie'}, inplace = True)`

In [80]: `moviebasics_df.head()`

Out[80]:

	<b>movie_id</b>	<b>movie</b>	<b>original_title</b>	<b>start_year</b>	<b>runtime_minutes</b>	<b>genres</b>
<b>0</b>	tt0063540	Sunghursh	Sunghursh	2013	175.000000	Action,Crime,Drama

	<b>movie_id</b>	<b>movie</b>	<b>original_title</b>	<b>start_year</b>	<b>runtime_minutes</b>	<b>genres</b>
<b>1</b>	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.000000	Biography,Drama
<b>2</b>	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.000000	Drama
<b>3</b>	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	86.186126	Comedy,Drama
<b>4</b>	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.000000	Comedy,Drama,Fantasy

```
In [81]: movies_and_budgets = budgets_df.merge(moviebasics_df, on = "movie", how = "inner")
```

```
In [83]: movies_and_budgets
```

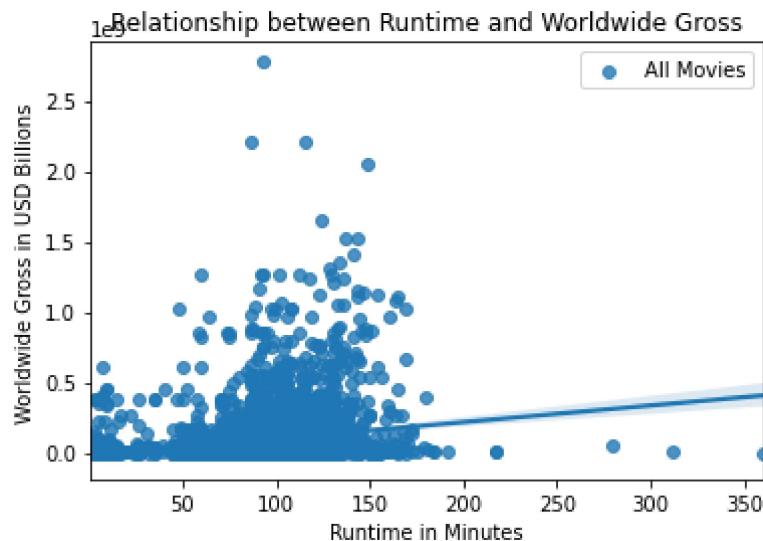
	<b>id</b>	<b>release_date</b>	<b>movie</b>	<b>production_budget</b>	<b>domestic_gross</b>	<b>worldwide_gross</b>	<b>budget_ratio</b>
<b>0</b>	1	Dec 18, 2009	Avatar	425000000.0	760507625.0	2.776345e+09	>5
			Pirates of the Caribbean: On Stranger Tides				
<b>1</b>	2	May 20, 2011	Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09	>5
<b>2</b>	3	Jun 7, 2019	Dark Phoenix	350000000.0	42762350.0	1.497624e+08	>5
<b>3</b>	4	May 1, 2015	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09	>5
<b>4</b>	7	Apr 27, 2018	Avengers: Infinity War	300000000.0	678815482.0	2.048134e+09	>5
...	...	...	...	...	...	...	...
<b>3738</b>	67	Apr 28, 2006	Clean	10000.0	138711.0	1.387110e+05	<
<b>3739</b>	68	Jul 6, 2001	Cure	10000.0	94596.0	9.459600e+04	<
<b>3740</b>	73	Jan 13, 2012	Newlyweds	9000.0	4584.0	4.584000e+03	<
<b>3741</b>	78	Dec 31, 2018	Red 11	7000.0	0.0	0.000000e+00	<
<b>3742</b>	81	Sep 29, 2015	A Plague So Pleasant	1400.0	0.0	0.000000e+00	<

3743 rows × 13 columns



```
In [84]: #analysing the joined dataframe
sns.regplot(x = 'runtime_minutes', y = 'worldwide_gross',
             label = 'All Movies', data = movies_and_budgets)
plt.xlabel('Runtime in Minutes')
plt.ylabel('Worldwide Gross in USD Billions')
plt.title('Relationship between Runtime and Worldwide Gross')
```

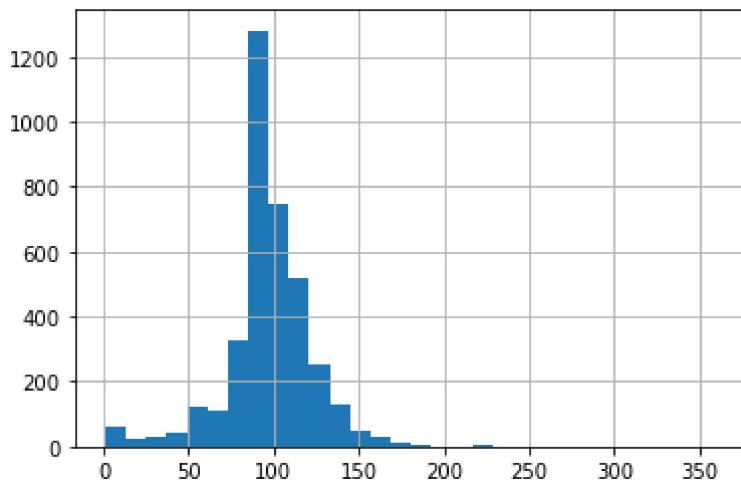
```
plt.legend()  
plt.show()
```



```
In [86]: movies_and_budgets['runtime_minutes'].mean()
```

```
Out[86]: 96.00528417900202
```

```
In [87]: movies_and_budgets["runtime_minutes"].hist(bins = 30);
```



## #Recommendation(Runtime)

There does not seem to be any correlation between runtime and worldwide gross. The safer option would be to aim for the average runtime which is 100 minutes