



New-Movie-Studio-Name / Movie Studios Report.ipynb

 protigen34 Add files via upload

[History](#)

 1 contributor

6763 lines (6763 sloc) | 343 KB

## Final Project Submission

- Student name: Charlie Jin
- Student pace: Full time
- Scheduled project review date/time: Mar 10
- Instructor name: Daniel
- Blog post URL:

# Microsoft's Movies Report

## Overview

My project analyzes movies earnings and box office results to reach a necessary conclusion. Microsoft needs to know that their investment in Flatiron Pictures Studios is a profitable one. To do this, my project needs to reach several milestones and conclusions to come up with three recommendations in order for Microsoft to believe in the possibility of studio profitability. Using open sources like IMDB, rottentomatoes, boxofficemojo a comprehensive analytical and mathematical report can and will be done.

## Business Problem

Microsoft is not confident in their investment in Flatiron Pictures Studio. Or they don't know what the plan of action is and the next steps to take. How to lead them in the right path.

## Data Understanding

### Part I: Creating SQL Dataframe

-Merge all the SQL schemas. There are 7 provided.

```
In [ ]: # Your code here - remember to use markdown cells for comments as well!
```

```
In [4]: import pandas as pd
import os
```

```
In [5]: import pandas as pd
import numpy as np
import sqlite3 as sql
```

```
In [6]: con = sql.connect(r"C:\Users\somep\Documents\New-Movie-Studio-Name\im.db")
```

```
In [7]: database = "im.db"
connection = sql.connect(database)
```

```
connection = sql.connect(database)
```

```
In [8]: query = '''SELECT * FROM im.db'''
```

```
In [9]: schema_df = pd.read_sql("""
SELECT *
FROM sqlite_master

""", con)

schema_df
```

```
Out[9]:
```

	type	name	tbl_name	rootpage	sql
0	table	movie_basics	movie_basics	2	CREATE TABLE "movie_basics" (\n"movie_id" TEXT...
1	table	directors	directors	3	CREATE TABLE "directors" (\n"movie_id" TEXT,\n...
2	table	known_for	known_for	4	CREATE TABLE "known_for" (\n"person_id" TEXT,\n...
3	table	movie_akas	movie_akas	5	CREATE TABLE "movie_akas" (\n"movie_id" TEXT,\n...
4	table	movie_ratings	movie_ratings	6	CREATE TABLE "movie_ratings" (\n"movie_id" TEX...
5	table	persons	persons	7	CREATE TABLE "persons" (\n"person_id" TEXT,\n ...
6	table	principals	principals	8	CREATE TABLE "principals" (\n"movie_id" TEXT,\n...
7	table	writers	writers	9	CREATE TABLE "writers" (\n"movie_id" TEXT,\n ...

```
In [10]: print(schema_df['sql'].iloc[0])
```

```
CREATE TABLE "movie_basics" (
"movie_id" TEXT,
"primary_title" TEXT,
"original_title" TEXT,
"start_year" INTEGER,
"runtime_minutes" REAL,
"genres" TEXT
)
```

```
In [11]: #import sqlite3
#conn = sqlite3.connect('im.db')
#c = conn.cursor()

#c.execute("""create table emp(movie_basics,directors,known_for,movie_akas,movie_ratings,
#conn.commit()
```

```
In [12]: movie_basics_df = pd.read_sql("""
SELECT *
FROM movie_basics

""", con)

movie_basics_df.head()
```

```
Out[12]:
```

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama

1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy

```
In [13]: directors_df = pd.read_sql("""
SELECT *
FROM directors
""", con)
directors_df.head()
```

Out[13]:

	movie_id	person_id
0	tt0285252	nm0899854
1	tt0462036	nm1940585
2	tt0835418	nm0151540
3	tt0835418	nm0151540
4	tt0878654	nm0089502

```
In [14]: known_for_df = pd.read_sql("""
SELECT *
FROM known_for
""", con)
known_for_df.head()
```

Out[14]:

	person_id	movie_id
0	nm0061671	tt0837562
1	nm0061671	tt2398241
2	nm0061671	tt0844471
3	nm0061671	tt0118553
4	nm0061865	tt0896534

```
In [15]: movie_akas_df = pd.read_sql("""
SELECT *
FROM movie_akas
""", con)
movie_akas_df.head()
```

Out[15]:

	movie_id	ordering	title	region	language	types	attributes	is_original_title
0	tt0369610	10	Лужнецкий свет	RG	bg	None	None	0.0

	movie_id	title	country	language	genre	imdbDisplay	short title	rating
1	tt0369610	Jurashikku warudo	JP	None	imdbDisplay	None		0.0
2	tt0369610	Jurassic World: O Mundo dos Dinossauros	BR	None	imdbDisplay	None		0.0
3	tt0369610	O Mundo dos Dinossauros	BR	None	None	short title		0.0
4	tt0369610	Jurassic World	FR	None	imdbDisplay	None		0.0

```
In [16]: movie_ratings_df = pd.read_sql("""
SELECT *
FROM movie_ratings
""", con)
movie_ratings_df.head()
```

Out[16]:

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

```
In [17]: persons_df = pd.read_sql("""
SELECT *
FROM persons
""", con)
persons_df.head()
```

Out[17]:

	person_id	primary_name	birth_year	death_year	primary_profession
0	nm0061671	Mary Ellen Bauder	NaN	NaN	miscellaneous,production_manager,producer
1	nm0061865	Joseph Bauer	NaN	NaN	composer,music_department,sound_department
2	nm0062070	Bruce Baum	NaN	NaN	miscellaneous,actor,writer
3	nm0062195	Axel Baumann	NaN	NaN	camera_department,cinematographer,art_department
4	nm0062798	Pete Baxter	NaN	NaN	production_designer,art_department,set_decorator

```
In [18]: principals_df = pd.read_sql("""
SELECT *
FROM principals
""", con)
principals_df.head()
```

Out[18]:

	movie_id	ordering	person_id	category	job	characters
0	tt0111414	1	nm0246005	actor	None	["The Man"]
1	tt0111414	2	nm0398271	director	None	None
2	tt0111414	3	nm3739909	producer	producer	None
3	tt0323808	10	nm0059247	editor	None	None
4	tt0323808	1	nm3579312	actress	None	["Beth Boothby"]

In [19]:

```
writers_df = pd.read_sql("""
SELECT *
FROM writers

""", con)

writers_df.head()
```

Out[19]:

	movie_id	person_id
0	tt0285252	nm0899854
1	tt0438973	nm0175726
2	tt0438973	nm1802864
3	tt0462036	nm1940585
4	tt0835418	nm0310087

In [20]:

```
movie_basics_directors = pd.merge(movie_basics_df, directors_df)
movie_basics_directors.columns
```

Out[20]: Index(['movie\_id', 'primary\_title', 'original\_title', 'start\_year',  
'runtime\_minutes', 'genres', 'person\_id'],  
dtype='object')

In [21]:

```
movie_basics_directors.head()
```

Out[21]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	person_id
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama	nm0712540
1	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama	nm0712540
2	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama	nm0712540
3	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama	nm0712540
4	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama	nm0002411

In [22]:

```
MOVIEDIRECTORSRATINGS = pd.merge(movie_basics_directors, movie_ratings_df)
MOVIEDIRECTORSRATINGS.columns
```

Out[22]: Index(['movie\_id', 'primary\_title', 'original\_title', 'start\_year',  
'runtime\_minutes', 'genres', 'person\_id', 'averagerating', 'numvotes'],  
dtype='object')

In [23]:

```
MOVIEDIRECTORSRATINGS.head()
```

Out[23]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	person_id	ave
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action, Crime, Drama	nm0712540	
1	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action, Crime, Drama	nm0712540	
2	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action, Crime, Drama	nm0712540	
3	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action, Crime, Drama	nm0712540	
4	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography, Drama	nm0002411	

In [24]:

```
MOVIEDIRECTORSRATINGS2 = pd.merge(MOVIEDIRECTORSRATINGS, known_for_df)
MOVIEDIRECTORSRATINGS2.columns
```

Out[24]:

```
Index(['movie_id', 'primary_title', 'original_title', 'start_year',
      'runtime_minutes', 'genres', 'person_id', 'averagerating', 'numvotes'],
      dtype='object')
```

In [25]:

```
MOVIEDIRECTORSRATINGS2.head()
```

Out[25]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	person_id	ave
0	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy, Drama, Fantasy	nm0765384	
1	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy, Drama, Fantasy	nm0765384	
2	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy, Drama, Fantasy	nm0749914	
3	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy, Drama, Fantasy	nm0749914	
4	tt0112502	Bigfoot	Bigfoot	2017	NaN	Horror, Thriller	nm6883878	

In [26]:

```
MOVIEDIRECTORSRATINGS3 = pd.merge(MOVIEDIRECTORSRATINGS2, movie_akas_df)
MOVIEDIRECTORSRATINGS3.columns
```

Out[26]:

```
Index(['movie_id', 'primary_title', 'original_title', 'start_year',
      'runtime_minutes', 'genres', 'person_id', 'averagerating', 'numvotes',
      'ordering', 'title', 'region', 'language', 'types', 'attributes',
      'is_original_title'],
      dtype='object')
```

In [27]:

```
MOVIEDIRECTORSRATINGS3.head()
```

Out[27]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	person_id	ave
0	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy, Drama, Fantasy	nm0765384	

1	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	nm0765384
2	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	nm0765384
3	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	nm0765384
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	nm0765384

```
In [28]: MOVIEDIRECTORSRATINGS4 = pd.merge(MOVIEDIRECTORSRATINGS3, persons_df)
MOVIEDIRECTORSRATINGS4.columns
```

```
Out[28]: Index(['movie_id', 'primary_title', 'original_title', 'start_year',
               'runtime_minutes', 'genres', 'person_id', 'averagerating', 'numvotes',
               'ordering', 'title', 'region', 'language', 'types', 'attributes',
               'is_original_title', 'primary_name', 'birth_year', 'death_year',
               'primary_profession'],
              dtype='object')
```

```
In [29]: MOVIEDIRECTORSRATINGS4.head()
```

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	person_id
0	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	nm0765384
1	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	nm0765384
2	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	nm0765384
3	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	nm0765384
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	nm0765384

```
In [30]: MOVIEDIRECTORSRATINGS5 = pd.merge(MOVIEDIRECTORSRATINGS4, principals_df)
MOVIEDIRECTORSRATINGS5.columns
```

```
Out[30]: Index(['movie_id', 'primary_title', 'original_title', 'start_year',
               'runtime_minutes', 'genres', 'person_id', 'averagerating', 'numvotes',
               'ordering', 'title', 'region', 'language', 'types', 'attributes',
               'is_original_title', 'primary_name', 'birth_year', 'death_year',
               'primary_profession', 'category', 'job', 'characters'],
              dtype='object')
```

```
In [31]: ..
```



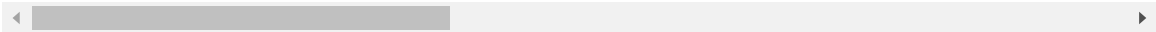
Out[30]:

```
MOVIEDIRECTORSRATINGS5.head()
```

Out[31]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres	person_id
0	tt1928329	Lines of Wellington	Linhas de Wellington	2012	151.0	Drama,History,War	nm0765384
1	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	nm0749914
2	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy	nm0749914
3	tt1236371	Mysteries of Lisbon	Mistérios de Lisboa	2010	272.0	Drama,Mystery,Romance	nm0749914
4	tt1236371	Mysteries of Lisbon	Mistérios de Lisboa	2010	272.0	Drama,Mystery,Romance	nm0749914

5 rows × 23 columns



In [107]:

```
MOVIEDIRECTORSRATINGS6 = pd.merge(MOVIEDIRECTORSRATINGS5, writers_df)
MOVIEDIRECTORSRATINGS6.columns
```

Out[107]:

```
Index(['movie_id', 'primary_title', 'original_title', 'start_year',
      'runtime_minutes', 'genres', 'person_id', 'averagerating', 'numvotes',
      'ordering', 'title', 'region', 'language', 'types', 'attributes',
      'is_original_title', 'primary_name', 'birth_year', 'death_year',
      'primary_profession', 'category', 'job', 'characters'],
      dtype='object')
```

## Part II: Merging SQL created Dataframe with CSV files

- Once all the SQL schemas have been merged, now merge the final SQL dataset with the CSV files provided.
- Drop all the columns that are not needed or are completely or nearly blank.
- Drop duplicate rows.
- Rename some columns that are the same.

In [108]:

```
MOVIEDIRECTORSRATINGS6.drop(['movie_id', 'original_title', 'person_id', 'types', 'ordering'])
```

In [109]:

```
MOVIEDIRECTORSRATINGS6.rename(columns = {'primary_title':'title','start_year':'year', 'runtime_minutes':'runtime'})
```

In [110]:

```
MOVIEDIRECTORSRATINGS6.drop_duplicates(inplace=True)
```

In [111]:

```
MOVIEDIRECTORSRATINGS6.head()
```

Out[111]:

	title	year	runtime	genre	average rating	total votes	region	language	primary_name
0	The Wandering Soap Opera	2017	80.0	Comedy,Drama,Fantasy	6.5	110	BR	None	Raquel Bortolotto

4	Joe Finds Grace	2017	83.0	Adventure,Animation,Comedy	8.1	263	CA	None	Anthor Harrisc
5	Pál Adrienn	2010	136.0	Drama	6.8	451	SE	None	Ágnes Kocs
7	Life's a Beach	2012	100.0	Comedy	3.9	219	None	None	Tony Vita
8	Second Coming	2012	95.0	None	5.5	20	US	None	Darre Campbe



```
In [112... tablefile1 = pd.read_csv(r"C:\Users\somep\Documents\New-Movie-Studio-Name\bom.movie_gross
```

```
In [113... tablefile1.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 [114... tablefile2=pd.read_table(r"C:\Users\somep\Documents\New-Movie-Studio-Name\rt.movie_info.t
tablefile2.head()
```

	id	synopsis	rating	genre	director	writer	theater_date	dvd_date
0	1	This gritty, fast-paced, and innovative police...	R	Action and Adventure Classics Drama	William Friedkin	Ernest Tidyman	Oct 9, 1971	Sep 25, 2001
1	3	New York City, not-too-distant-future: Eric Pa...	R	Drama Science Fiction and Fantasy	David Cronenberg	David Cronenberg Don DeLillo	Aug 17, 2012	Jan 1, 2013
2	5	Illeana Douglas delivers a superb performance ...	R	Drama Musical and Performing Arts	Allison Anders	Allison Anders	Sep 13, 1996	Apr 18, 2000
3	6	Michael Douglas runs afoul of a treacherous su...	R	Drama Mystery and Suspense	Barry Levinson	Paul Attanasio Michael Crichton	Dec 9, 1994	Aug 27, 1997
4	7	NaN	NR	Drama Romance	Rodney Bennett	Giles Cooper	NaN	NaN

```
In [115...
tablefile2.rename(columns = {'theater_date':'release date'}, inplace=True)

In [116...
tablefile2.drop(['id'], axis = 1, inplace=True)

In [117...
tablefile2.head()
```

Out[117...

	synopsis	rating	genre	director	writer	release date	dvd_date	currency
0	This gritty, fast-paced, and innovative police...	R	Action and Adventure Classics Drama	William Friedkin	Ernest Tidyman	Oct 9, 1971	Sep 25, 2001	NaN
1	New York City, not-too-distant-future: Eric Pa...	R	Drama Science Fiction and Fantasy	David Cronenberg	David Cronenberg Don DeLillo	Aug 17, 2012	Jan 1, 2013	\$
2	Illeana Douglas delivers a superb performance ...	R	Drama Musical and Performing Arts	Allison Anders	Allison Anders	Sep 13, 1996	Apr 18, 2000	NaN
3	Michael Douglas runs afoul of a treacherous su...	R	Drama Mystery and Suspense	Barry Levinson	Paul Attanasio Michael Crichton	Dec 9, 1994	Aug 27, 1997	NaN
4	NaN	NR	Drama Romance	Rodney Bennett	Giles Cooper	NaN	NaN	NaN

```
In [118...
tablefile3 = pd.read_csv(r"C:\Users\somep\Documents\New-Movie-Studio-Name\tmdb.movies.csv")
tablefile3.head()
```

Out[118...

Unnamed: 0	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_
0	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	1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon
2	2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2
3	3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story

4

4

[28, 878, 12]

27205

en

Inception

27.920

2010-07-16

Inception

```
In [119... tablefile3.rename(columns = {'release_date':'release date', 'vote_average':'average rating
```

```
In [120... tablefile3.drop(['Unnamed: 0', 'genre_ids', 'id', 'original_title'], axis = 1, inplace=Tr
```

```
In [121... tablefile3.head()
```

Out[121...

	original_language	popularity	release date	title	average rating	total votes
0	en	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	7.7	10788
1	en	28.734	2010-03-26	How to Train Your Dragon	7.7	7610
2	en	28.515	2010-05-07	Iron Man 2	6.8	12368
3	en	28.005	1995-11-22	Toy Story	7.9	10174
4	en	27.920	2010-07-16	Inception	8.3	22186

```
In [122... tablefile4 = pd.read_csv(r"C:\Users\somep\Documents\New-Movie-Studio-Name\tn.movie_budget
tablefile4.head()
```

Out[122...

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

```
In [123... tablefile4.drop(['id'], axis = 1, inplace=True)
```

```
In [124... tablefile4.rename(columns = {'release_date':'release date', 'movie':'title'}, inplace=Tr
```

```
In [125... tablefile4.head()
```

Out[125...

	release date	title	production_budget	domestic_gross	worldwide_gross
0	Dec 18, 2009	Avatar	\$425,000,000	\$760,507,625	\$2,776,345,279
1	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	\$410,600,000	\$241,063,875	\$1,045,663,875

2	Jun 7, 2019	Dark Phoenix	\$350,000,000	\$42,762,350	\$149,762,350
3	May 1, 2015	Avengers: Age of Ultron	\$330,600,000	\$459,005,868	\$1,403,013,963
4	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	\$317,000,000	\$620,181,382	\$1,316,721,747

```
In [126... MOVIEDIRECTORSRATINGS7 = pd.merge(tablefile1, tablefile3)
MOVIEDIRECTORSRATINGS7.head()
```

Out[126...

	title	studio	domestic_gross	foreign_gross	year	original_language	popularity	release date	average rating
0	Toy Story 3	BV	415000000.0	652000000	2010	en	24.445	2010-06-17	7.7
1	Inception	WB	292600000.0	535700000	2010	en	27.920	2010-07-16	8.3
2	Shrek Forever After	P/DW	238700000.0	513900000	2010	en	15.041	2010-05-16	6.1
3	The Twilight Saga: Eclipse	Sum.	300500000.0	398000000	2010	en	20.340	2010-06-23	6.0
4	Iron Man 2	Par.	312400000.0	311500000	2010	en	28.515	2010-05-07	6.8

```
In [128... to_concat = [tablefile4, MOVIEDIRECTORSRATINGS7]
MOVIEDIRECTORSRATINGS8 = pd.concat(to_concat)
MOVIEDIRECTORSRATINGS8.head()
```

Out[128...

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

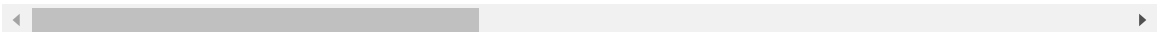
```
In [139... to_concat = [MOVIEDIRECTORSRATINGS6, MOVIEDIRECTORSRATINGS8]
```

```
MOVIEDIRECTORSRATINGS9 = pd.concat(to_concat)
MOVIEDIRECTORSRATINGS9.head()
```

Out[139...

	title	year	runtime	genre	average rating	total votes	region	language	primary_na
0	The Wandering Soap Opera	2017.0	80.0	Comedy,Drama,Fantasy	6.5	119.0	PL	None	Raoul F
4	Joe Finds Grace	2017.0	83.0	Adventure,Animation,Comedy	8.1	263.0	CA	None	Anthr Harri:
5	Pál Adrienn	2010.0	136.0	Drama	6.8	451.0	SE	None	Ágnes Ko
7	Life's a Beach	2012.0	100.0	Comedy	3.9	219.0	None	None	Tony Vit
8	Second Coming	2012.0	95.0	None	5.5	20.0	US	None	Dar Campl

5 rows × 23 columns



In [140...

```
excel_file = pd.ExcelWriter("FINAL DATASET.xlsx")
```

In [141...

```
MOVIEDIRECTORSRATINGS9.to_excel(excel_file)
```

In [142...

```
excel_file.save()
```

In [143...

```
FINALDATASETCLEANED = pd.read_csv(r"C:\Users\somep\Documents\New-Movie-Studio-Name\dsc-ph
```

In [144...

```
FINALDATASETCLEANED=pd.read_csv(r"C:\Users\somep\Documents\New-Movie-Studio-Name\dsc-phas
```

In [145...

```
FINALDATASETCLEANED.dtypes
```

Out[145...

Unnamed: 0	int64
title	object
year	float64
runtime	float64
genre	object
average rating	float64
total votes	float64
region	object
language	object
primary_name	object
birth_year	float64
death_year	float64
primary_profession	object
category	object
job	object
characters	object
release date	datetime64[ns]
production_budget	object
domestic_gross	object
foreign_gross	object
worldwide_gross	object

```
net_gain      object
studio        object
original_language  object
popularity    float64
dtype: object
```

### Part III: Data Cleaning

- change the date format display to a uniform display for all rows.
- remove the \$ and comma sign from box office gross in order to make column a number or float.
- drop unnecessary and no use columns.
- change all data types of "objects" to "floats."

```
In [146... FINALDATASETCLEANED['release date']=pd.to_datetime(FINALDATASETCLEANED['release date'], format='%m-%d-%Y')
FINALDATASETCLEANED['release date']=FINALDATASETCLEANED['release date'].dt.strftime('%d-%m-%Y')
```

```
In [147... FINALDATASETCLEANED.tail()
```

Out[147...]

	Unnamed: 0	title	year	runtime	genre	average rating	total votes	region	language	primary_name	...
21380	2698	The Escape	2018.0	NaN	NaN	7.0	1.0	NaN	NaN	NaN	...
21381	2699	The Escape	2018.0	NaN	NaN	6.6	10.0	NaN	NaN	NaN	...
21382	2700	Souvenir	2018.0	NaN	NaN	5.8	14.0	NaN	NaN	NaN	...
21383	2701	The Quake	2018.0	NaN	NaN	6.7	81.0	NaN	NaN	NaN	...
21384	2702	An Actor Prepares	2018.0	NaN	NaN	6.5	10.0	NaN	NaN	NaN	...

5 rows × 25 columns

```
In [148... FINALDATASETCLEANED.drop(['Unnamed: 0', 'region', 'language', 'primary_name', 'birth_year', 'gross', 'production_budget', 'country', 'studio', 'original_language', 'popularity', 'net_gain', 'runtime', 'genre', 'average rating', 'total votes', 'release date', 'primary_name'])
```

```
In [149... FINALDATASETCLEANED.head()
```

Out[149...]

	title	year	runtime	genre	average rating	total votes	release date	production_budget	country
0	The Wandering Soap Opera	2017.0	80.0	Comedy,Drama,Fantasy	6.5	119.0	NaN	NaN	NaN
1	Joe Finds Gross	2017.0	83.0	Adventure,Animation,Comedy	8.1	263.0	NaN	NaN	NaN

	Grace								
2	Pál Adrienn	2010.0	136.0	Drama	6.8	451.0	NaN	NaN	
3	Life's a Beach	2012.0	100.0	Comedy	3.9	219.0	NaN	NaN	
4	Second Coming	2012.0	95.0	NaN	5.5	20.0	NaN	NaN	



In [150... FINALDATASETCLEANED['production\_budget']=FINALDATASETCLEANED['production\_budget'].str.replace

C:\Users\sompe\AppData\Local\Temp\ipykernel\_4624\2341965880.py:1: FutureWarning: The default value of regex will change from True to False in a future version.

FINALDATASETCLEANED['production\_budget']=FINALDATASETCLEANED['production\_budget'].str.replace(r'\W', "")

In [151... FINALDATASETCLEANED['domestic\_gross']=FINALDATASETCLEANED['domestic\_gross'].str.replace

C:\Users\sompe\AppData\Local\Temp\ipykernel\_4624\2838814049.py:1: FutureWarning: The default value of regex will change from True to False in a future version.

FINALDATASETCLEANED['domestic\_gross']=FINALDATASETCLEANED['domestic\_gross'].str.replace(r'\W', "")

In [152... FINALDATASETCLEANED['foreign\_gross']=FINALDATASETCLEANED['foreign\_gross'].str.replace

C:\Users\sompe\AppData\Local\Temp\ipykernel\_4624\3381301098.py:1: FutureWarning: The default value of regex will change from True to False in a future version.

FINALDATASETCLEANED['foreign\_gross']=FINALDATASETCLEANED['foreign\_gross'].str.replace(r'\W', "")

In [153... FINALDATASETCLEANED['worldwide\_gross']=FINALDATASETCLEANED['worldwide\_gross'].str.replace

C:\Users\sompe\AppData\Local\Temp\ipykernel\_4624\2466272076.py:1: FutureWarning: The default value of regex will change from True to False in a future version.

FINALDATASETCLEANED['worldwide\_gross']=FINALDATASETCLEANED['worldwide\_gross'].str.replace(r'\W', "")

In [154... FINALDATASETCLEANED['net gain']=FINALDATASETCLEANED['net gain'].str.replace(r'\W', "")

C:\Users\sompe\AppData\Local\Temp\ipykernel\_4624\171332.py:1: FutureWarning: The default value of regex will change from True to False in a future version.

FINALDATASETCLEANED['net gain']=FINALDATASETCLEANED['net gain'].str.replace(r'\W', "")

In [155... FINALDATASETCLEANED['production\_budget'] = FINALDATASETCLEANED['production\_budget'].astype

In [156... FINALDATASETCLEANED['domestic\_gross'] = FINALDATASETCLEANED['domestic\_gross'].astype(float)

In [157... FINALDATASETCLEANED['foreign\_gross'] = FINALDATASETCLEANED['foreign\_gross'].astype(float)

In [158... FINALDATASETCLEANED['worldwide\_gross'] = FINALDATASETCLEANED['worldwide\_gross'].astype(float)

In [159... FINALDATASETCLEANED['net gain'] = FINALDATASETCLEANED['net gain'].astype(float)



```
In [160... excel_file = pd.ExcelWriter("FINAL DATASET CLEANED.xlsx")

In [161... FINALDATASETCLEANED.to_excel(excel_file)

In [162... excel_file.save()
```

## Part IV: Analysis

Trying to obtain the results and figures from the 4 main information columns: Production Budget, Domestic Gross, Worldwide Gross, and Net Gain.

```
In [163... myownprivatedataset = pd.read_csv(r"C:\Users\somep\Documents\New-Movie-Studio-Name\dsc-ph

In [165... myownprivatedataset.head()
```

Out[165...

	i»¿	title	year	runtime	genre	average rating	total votes	release date	production_budget	domestic_gross	f
0	0	Star Wars: The Force Awakens	2015.0	NaN	NaN	7.4	12641.0	18-Dec-2015	NaN	936700000.0	1
1	1	Star Wars: The Force Awakens	2015.0	NaN	NaN	7.4	12641.0	18-Dec-2015	NaN	936700000.0	1
2	2	Star Wars Ep. VII: The Force Awakens	NaN	NaN	NaN	NaN	NaN	18-Dec-2015	306000000.0	936662225.0	1
3	3	Avatar	NaN	NaN	NaN	NaN	NaN	18-Dec-2009	425000000.0	760507625.0	2
4	4	Black Panther	2018.0	NaN	NaN	5.1	11.0	18-Jan-2011	NaN	700100000.0	6

```
In [166... myownprivatedataset.dtypes
```

Out[166...

i»¿	int64
title	object
year	float64
runtime	float64
genre	object
average rating	float64
total votes	float64
release date	object
production_budget	float64
domestic_gross	float64
foreign_gross	float64

```
worldwide_gross    float64
net gain           float64
studio             object
original_language  object
popularity         float64
dtype: object
```

```
In [167... myownprivatedataset.nlargargest(10, columns=['production_budget'], keep='first')
```

Out[167...

	i>¿	title	year	runtime	genre	average rating	total votes	release date	production_budget	domestic_gross
3	3	Avatar	NaN	NaN	NaN	NaN	NaN	18-Dec-2009	425000000.0	760507625.0
207	207	Pirates of the Caribbean: On Stranger Tides	NaN	NaN	NaN	NaN	NaN	20-May-2011	410600000.0	241063875.0
2357	2357	Dark Phoenix	NaN	NaN	NaN	NaN	NaN	07-Jun-2019	350000000.0	42762350.0
27	27	Avengers: Age of Ultron	NaN	NaN	NaN	NaN	NaN	01-May-2015	330600000.0	459005868.0
15	15	Star Wars Ep. VIII: The Last Jedi	NaN	NaN	NaN	NaN	NaN	15-Dec-2017	317000000.0	620181382.0
2	2	Star Wars Ep. VII: The Force Awakens	NaN	NaN	NaN	NaN	NaN	18-Dec-2015	306000000.0	936662225.0
7	7	Avengers: Infinity War	NaN	NaN	NaN	NaN	NaN	27-Apr-2018	300000000.0	678815482.0
123	123	Pirates of the Caribbean: At Worlds End	NaN	NaN	NaN	NaN	NaN	24-May-2007	300000000.0	309420425.0
228	228	Justice League	NaN	NaN	NaN	NaN	NaN	17-Nov-2017	300000000.0	229024295.0
312	312	Spectre	NaN	NaN	NaN	NaN	NaN	06-Nov-2015	300000000.0	200074175.0

```
In [168... myownprivatedataset.dropna(subset = ['production_budget'], inplace=True)
```

```
In [169... myownprivatedataset.nlargest(10, columns=['domestic_gross'], keep='first')
```

Out[169...

	i>¿	title	year	runtime	genre	average rating	total votes	release date	production_budget	domestic_gross	fo
--	-----	-------	------	---------	-------	----------------	-------------	--------------	-------------------	----------------	----

2	2	Star Wars Ep. VII: The Force Awakens	NaN	NaN	NaN	NaN	NaN	18-Dec-2015	306000000.0	936662225.0	1.
3	3	Avatar	NaN	NaN	NaN	NaN	NaN	18-Dec-2009	425000000.0	760507625.0	2.
6	6	Black Panther	NaN	NaN	NaN	NaN	NaN	16-Feb-2018	200000000.0	700059566.0	6.
7	7	Avengers: Infinity War	NaN	NaN	NaN	NaN	NaN	27-Apr-2018	300000000.0	678815482.0	1.
9	9	Titanic	NaN	NaN	NaN	NaN	NaN	19-Dec-1997	200000000.0	659363944.0	1.
11	11	Jurassic World	NaN	NaN	NaN	NaN	NaN	12-Jun-2015	215000000.0	652270625.0	9.
12	12	The Avengers	NaN	NaN	NaN	NaN	NaN	04-May-2012	225000000.0	623279547.0	8.
15	15	Star Wars Ep. VIII: The Last Jedi	NaN	NaN	NaN	NaN	NaN	15-Dec-2017	317000000.0	620181382.0	6.
17	17	Incredibles 2	NaN	NaN	NaN	NaN	NaN	15-Jun-2018	200000000.0	608581744.0	6.
18	18	The Dark Knight	NaN	NaN	NaN	NaN	NaN	18-Jul-2008	185000000.0	533720947.0	4.



In [170...

```
myownprivatedataset.nlargest(10, columns=['worldwide_gross'], keep='first')
```

Out[170...

rank	id	title	year	runtime	genre	average rating	total votes	release date	production_budget	domestic_gross	for
3	3	Avatar	NaN	NaN	NaN	NaN	NaN	18-Dec-2009	425000000.0	760507625.0	2.0
9	9	Titanic	NaN	NaN	NaN	NaN	NaN	19-Dec-1997	200000000.0	659363944.0	1.5
2	2	Star Wars Ep. VII: The Force Awakens	NaN	NaN	NaN	NaN	NaN	18-Dec-2015	306000000.0	936662225.0	1.1
7	7	Avengers: Infinity War	NaN	NaN	NaN	NaN	NaN	27-Apr-2018	300000000.0	678815482.0	1.3
11	11	Jurassic World	NaN	NaN	NaN	NaN	NaN	12-Jun-2015	215000000.0	652270625.0	9.9

79	79	Furious 7	NaN	NaN	NaN	NaN	NaN	Apr-2015	190000000.0	353007020.0	1.1
12	12	The Avengers	NaN	NaN	NaN	NaN	NaN	04-May-2012	225000000.0	623279547.0	8.9
27	27	Avengers: Age of Ultron	NaN	NaN	NaN	NaN	NaN	01-May-2015	330600000.0	459005868.0	9.4
6	6	Black Panther	NaN	NaN	NaN	NaN	NaN	16-Feb-2018	200000000.0	700059566.0	6.4
62	62	Harry Potter and the Deathly Hallows: Part II	NaN	NaN	NaN	NaN	NaN	15-Jul-2011	125000000.0	381193157.0	9.6



In [171...

```
myownprivatedataset.nlargest(10, columns=['net gain'], keep='first')
```

Out[171...

rank	id	title	year	runtime	genre	average rating	total votes	release date	production_budget	domestic_gross	for
3	3	Avatar	NaN	NaN	NaN	NaN	NaN	18-Dec-2009	425000000.0	760507625.0	2.0
9	9	Titanic	NaN	NaN	NaN	NaN	NaN	19-Dec-1997	200000000.0	659363944.0	1.5
7	7	Avengers: Infinity War	NaN	NaN	NaN	NaN	NaN	27-Apr-2018	300000000.0	678815482.0	1.3
2	2	Star Wars Ep. VII: The Force Awakens	NaN	NaN	NaN	NaN	NaN	18-Dec-2015	306000000.0	936662225.0	1.1
11	11	Jurassic World	NaN	NaN	NaN	NaN	NaN	12-Jun-2015	215000000.0	652270625.0	9.9
79	79	Furious 7	NaN	NaN	NaN	NaN	NaN	03-Apr-2015	190000000.0	353007020.0	1.1
12	12	The Avengers	NaN	NaN	NaN	NaN	NaN	04-May-2012	225000000.0	623279547.0	8.9
62	62	Harry Potter and the Deathly Hallows: Part II	NaN	NaN	NaN	NaN	NaN	15-Jul-2011	125000000.0	381193157.0	9.6
6	6	Black Panther	NaN	NaN	NaN	NaN	NaN	16-Feb-2018	200000000.0	700059566.0	6.4
		Jurassic									

38	38	World: Fallen Kingdom	NaN	NaN	NaN	NaN	NaN	NaN	Jun- 2018	170000000.0	417719760.0	8.8
----	----	-----------------------------	-----	-----	-----	-----	-----	-----	--------------	-------------	-------------	-----



In [172...

```
myownprivatedataset.nsmallest(10, columns=['net gain'], keep='first')
```

Out[172...

	i>>	title	year	runtime	genre	average rating	total votes	release date	production_budget	domestic_gro
2357	2357	Dark Phoenix	NaN	NaN	NaN	NaN	NaN	07- Jun- 2019	350000000.0	42762350
8374	8374	Moonfall	NaN	NaN	NaN	NaN	NaN	31- Dec- 2020	150000000.0	(
3562	3562	Mars Needs Moms	NaN	NaN	NaN	NaN	NaN	11- Mar- 2011	150000000.0	21392758
5469	5469	Men in Black: International	NaN	NaN	NaN	NaN	NaN	14- Jun- 2019	110000000.0	3100000
4879	4879	Town & Country	NaN	NaN	NaN	NaN	NaN	27- Apr- 2001	105000000.0	6712451
5198	5198	The Adventures of Pluto Nash	NaN	NaN	NaN	NaN	NaN	16- Aug- 2002	100000000.0	4411102
8101	8101	Bright	NaN	NaN	NaN	NaN	NaN	13- Dec- 2017	90000000.0	(
8102	8102	Army of the Dead	NaN	NaN	NaN	NaN	NaN	31- Dec- 2019	90000000.0	(
8111	8111	Call of the Wild	NaN	NaN	NaN	NaN	NaN	21- Feb- 2020	82000000.0	(
4681	4681	The Promise	NaN	NaN	NaN	NaN	NaN	21- Apr- 2017	90000000.0	8224288



In [173...

```
myownprivatedataframe = pd.DataFrame({'title':['Avatar', 'Pirates of the Caribbean: On S',  
      'production_budget':['425000000', '410600000', '350000000',  
      'domestic_gross':['760507625', '241063875', '427623500',  
      'worldwide_gross':['2776345279', '1045663875', '1497000000',  
      'net gain':['2351345279', '635063875', '-200237650'],  
      print(myownprivatedataframe)
```

	title	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	Star Wars Ep. VIII: The Last Jedi	317000000	
5	Star Wars Ep. VII: The Force Awakens	306000000	
6	Avengers: Endgame	300000000	

```

6               Avengers: infinity war           300000000
7   Pirates of the Caribbean: At Worlds End      300000000
8               Justice League                   300000000
9               Spectre                          300000000
10              Black Panther                    200000000
11              Titanic                          200000000
12              Jurassic World                   215000000
13              The Avengers                     225000000
14              Incredibles 2                   200000000
15              The Dark Knight                  185000000
16              Furious 7                       190000000
17   Harry Potter and the Deathly Hallows: Part II 125000000
18              Jurassic World: Fallen Kingdom   170000000

```

```

      domestic_gross worldwide_gross    net gain
0      760507625      2776345279  2351345279
1      241063875      1045663875   635063875
2       42762350       149762350  -200237650
3      459005868      1403013963  1072413963
4      620181382      1316721747   999721747
5      936662225      2053311220  1747311220
6      678815482      2048134200  1748134200
7      309420425       963420425   663420425
8      229024295       655945209   355945209
9      200074175       879620923   579620923
10     700059566      1348258224  1148258224
11     659363944      2208208395  2008208395
12     652270625      1648854864  1433854864
13     623279547      1517935897  1292935897
14     608581744      1242520711  1042520711
15     533720947      1001996207   816996207
16     353007020      1518722794  1328722794
17     381193157      1341693157  1216693157
18     417719760      1305772799  1135772799

```

## Part V: Possible Useful Charts and Results

### bar charts:

(These necessary figures can be obtained here. These are charts that could possibly be useful in the final study.)

-production\_budget

-domestic/worldwide gross

-net gain

-the combination of two or more of these figures together

In [230...

```
print(myownprivatedataset[['title', 'production_budget', 'domestic_gross', 'worldwide_gross']])
```

```

      title  production_budget  domestic_gross \
2   Star Wars Ep. VII: The Force Awakens      306000000.0    936662225.0
3               Avatar              425000000.0    760507625.0
6               Black Panther              200000000.0    700059566.0
7   Avengers: Infinity War              300000000.0    678815482.0
9               Titanic              200000000.0    659363944.0
...
8459      Lunchtime Heroes              100000.0         0.0
8460              Open Secret              100000.0         0.0
8461      The Night Visitor              100000.0         0.0
8462              Tiger Orange              100000.0         0.0

```

```

8463          family motocross          10000.0          0.0

      worldwide_gross      net gain
2      2.053311e+09  1.747311e+09
3      2.776345e+09  2.351345e+09
6      1.348258e+09  1.148258e+09
7      2.048134e+09  1.748134e+09
9      2.208208e+09  2.008208e+09
...      ...      ...
8459      0.000000e+00 -1.000000e+05
8460      0.000000e+00 -1.000000e+05
8461      0.000000e+00 -1.000000e+05
8462      0.000000e+00 -1.000000e+05
8463      0.000000e+00 -1.000000e+04

```

[5782 rows x 5 columns]

In [231...

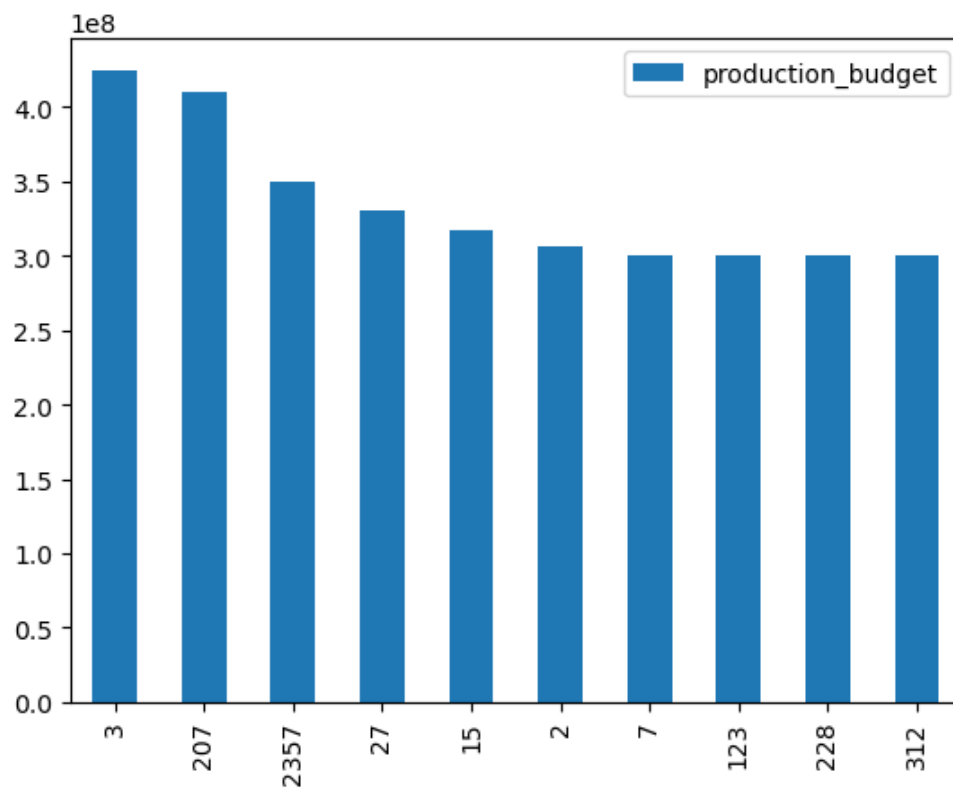
```
budget = myownprivatedataset[['title', 'production_budget']]
```

In [232...

```
budgetdata = budget.nlargest(10, columns=['production_budget'], keep='first')
```

In [214...

```
budgetplot=budgetdata.plot(kind="bar")
```



In [215...

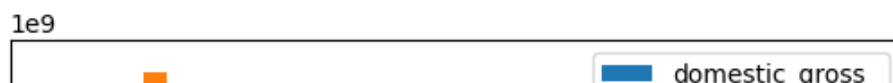
```
gross = myownprivatedataset[['title', 'domestic_gross', 'worldwide_gross']]
```

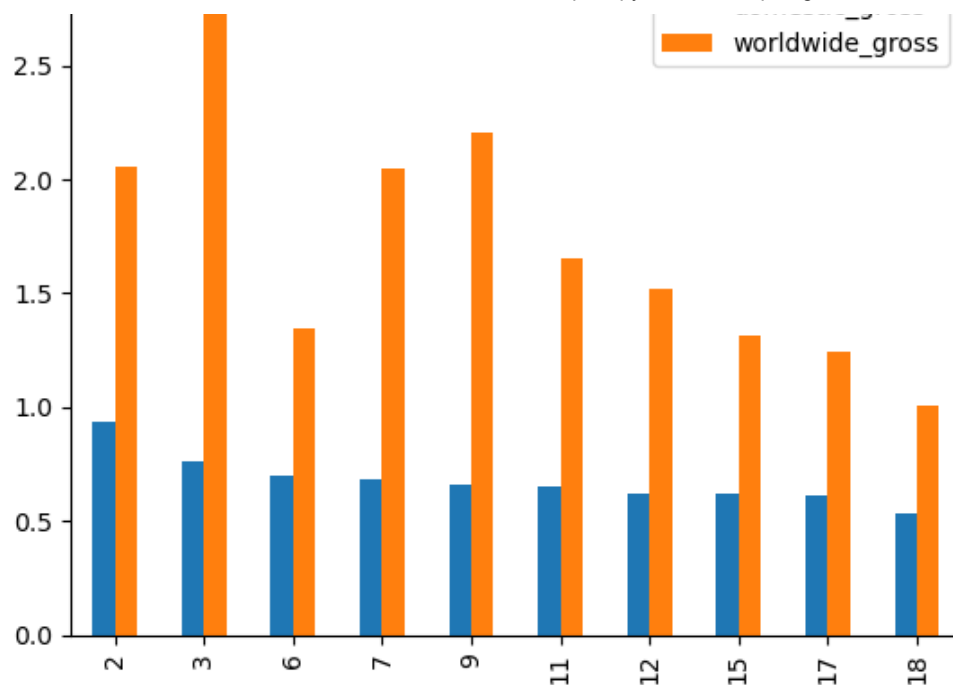
In [216...

```
grossdata = gross.nlargest(10, columns=['domestic_gross'], keep='first')
```

In [217...

```
grossplot=grossdata.plot(kind="bar")
```

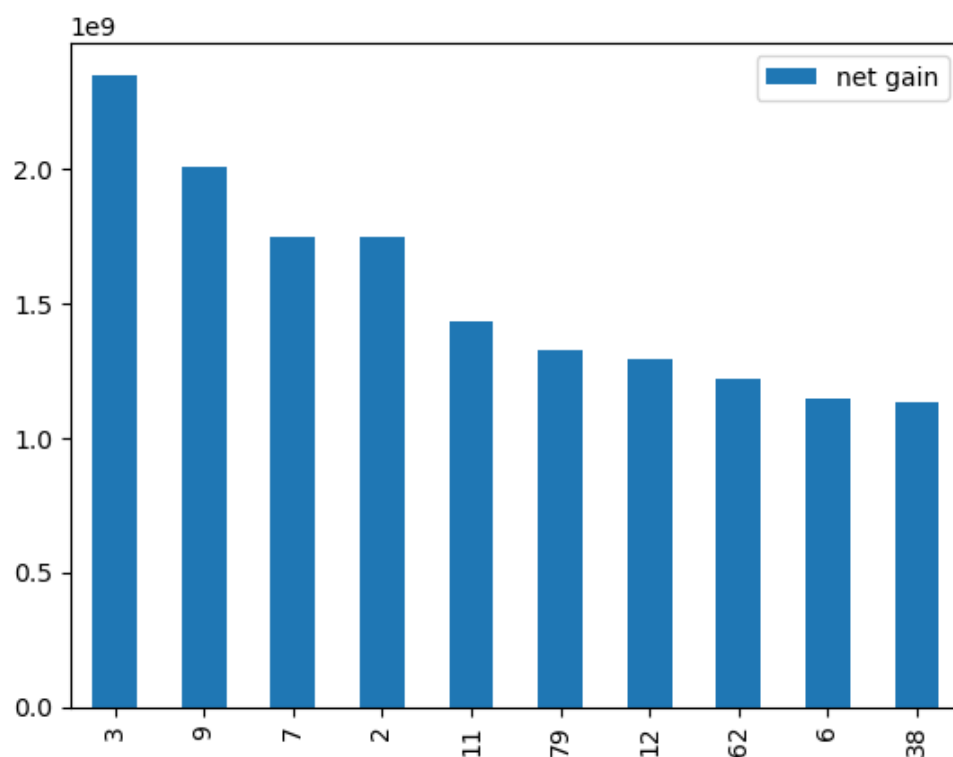




```
In [218... netgain = myownprivatedataset[['title', 'net gain']]
```

```
In [219... netgaindata = netgain.nlargest(10, columns=['net gain'], keep='first')
```

```
In [220... netgainplot=netgaindata.plot(kind="bar")
```



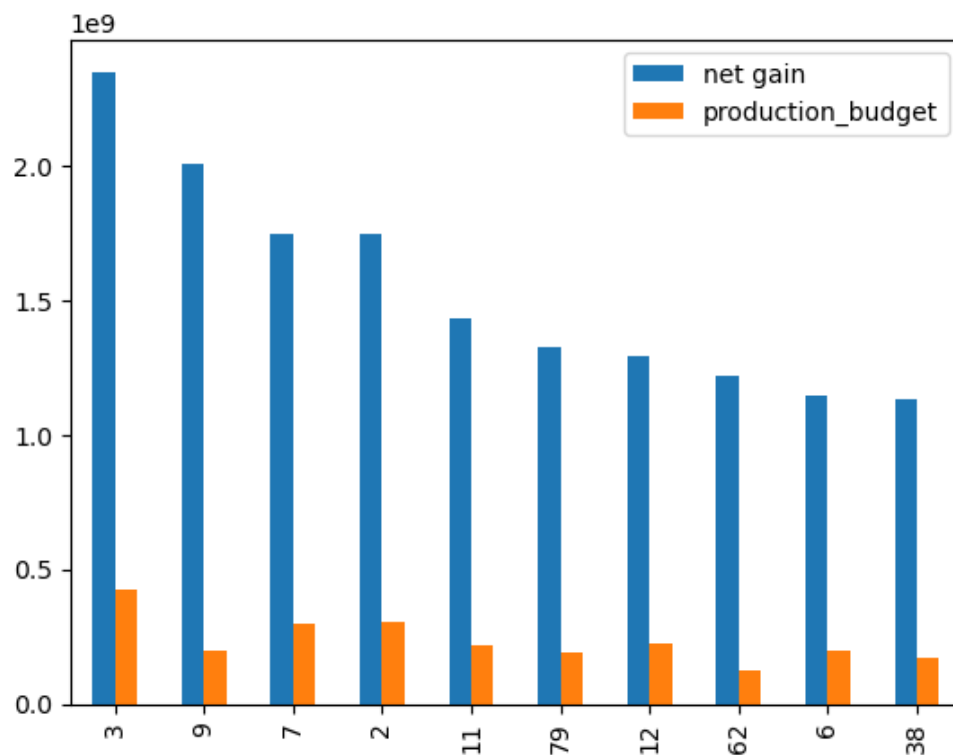
```
In [221... netgain2 = myownprivatedataset[['title', 'net gain', 'production_budget']]
```

```
In [222... netgain2data = netgain2.nlargest(10, columns=['net gain'], keep='first')
```



In [223...

```
netgain2plot=netgain2data.plot(kind="bar")
```



In [224...

```
netgainnegative = myownprivatedataset[['title', 'net gain']]
```

In [225...

```
netgainnegativedata = netgainnegative.nsmallest(10, columns=['net gain'], keep='first')
```

In [226...

```
netgainnegativeplot=netgainnegativedata.plot(kind="bar")
```

1e9