DATA UNDERSTANDING

For this project we will be having 6 data sources:

1. `bom.movie_gross.csv` : This is a file that contains information regarding a movies name, both domestic and foreign gross amount and the year it was released
2. `im.db` :This is a zipped folder that contains a database called `im` which contains data regarding a movie. This data includes what category it falls under, the direcctor, actors involved, movie ratings, running time and many more
3. `rt.movie_info.tsv` : This file has information the movie rating for theatrical releases e.g. 'R' 'PG 13' e.t.c. it also tells more about the movie like genre, director, theatrical release date, dvd release date, runtime, box office gross and studio
4. `rt.reviews.tsv` : This is a list of reviews from reviewers and and the ratings they gave, it also gives info on who published the review
5. `tmdb.movies.csv` : Here we have genre ids, movie titles, language the movie is in, its popularity, release date,vote count and the average vote
6. `tn.movie_budgets.csv` : This file contains a movies production budget, domestic gross and worlwide gross

DATA PREPARATION

Now we want to take a closer look at the data sources above

In [2]:
```
#We start by importing the relevant libraries that we would use to explore the data
import pandas as pd
import sqlite3
import numpy as np
import matplotlib.pyplot as plt
import zipfile
import io
import ast
from sklearn.preprocessing import MinMaxScaler
from sqlalchemy import create_engine
```

Now we want to load the data sources below

In [3]:
```
mg = pd.read_csv(r"zippedData\bom.movie_gross.csv.gz")
mg
```

Out[3]:

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

3387 rows × 5 columns

In [248…

```
# Loading 'rt.movie_info.tsv'
mi = pd.read_csv(r"zippedData\rt.movie_info.tsv.gz",  sep = '\t', encoding='ISO-885
mi
```

Out[248...

	id	synopsis	rating	genre	director	write
0	1	This gritty, fast-paced, and innovative police...	R	Action and Adventure|Classics|Drama	William Friedkin	Ernest Tidyma
1	3	New York City, not-too-distant-future: Eric Pa...	R	Drama|Science Fiction and Fantasy	David Cronenberg	Davi Cronenberg|Do DeLill
2	5	Illeana Douglas delivers a superb performance ...	R	Drama|Musical and Performing Arts	Allison Anders	Allison Ander
3	6	Michael Douglas runs afoul of a treacherous su...	R	Drama|Mystery and Suspense	Barry Levinson	Pau Attanasio|Michae Crichto
4	7	NaN	NR	Drama|Romance	Rodney Bennett	Giles Coope
...	...	...	...	...	...	.
1555	1996	Forget terrorists or hijackers -- there's a ha...	R	Action and Adventure|Horror|Mystery and Suspense	NaN	Nal
1556	1997	The popular Saturday Night Live sketch was exp...	PG	Comedy|Science Fiction and Fantasy	Steve Barron	Terry Turner|Tor Davis|Da Aykroyd|Bonni Turne
1557	1998	Based on a novel by Richard Powell, when the l...	G	Classics|Comedy|Drama|Musical and Performing Arts	Gordon Douglas	Nal
1558	1999	The Sandlot is a coming-of-age story about a g...	PG	Comedy|Drama|Kids and Family|Sports and Fitness	David Mickey Evans	David Micke Evans|Robe Gunte
1559	2000	Suspended from the	R	Action and Adventure|Art House and Internation...	NaN	Luc Besso

id	synopsis	rating	genre	director	write
	force, Paris cop Hubert is ...				

1560 rows × 12 columns

In [249…

```
# Load and view `rt.reviews.tsv`
mr = pd.read_csv(r"zippedData\rt.reviews.tsv.gz", sep = '\t', encoding='ISO-8859-1'
mr
```

Out[249...

	id	review	rating	fresh	critic	top_critic	publisher	d
0	3	A distinctly gallows take on contemporary fina...	3/5	fresh	PJ Nabarro	0	Patrick Nabarro	Novem 10, 2
1	3	It's an allegory in search of a meaning that n...	NaN	rotten	Annalee Newitz	0	io9.com	May 2
2	3	... life lived in a bubble in financial dealin...	NaN	fresh	Sean Axmaker	0	Stream on Demand	Januar 2
3	3	Continuing along a line introduced in last yea...	NaN	fresh	Daniel Kasman	0	MUBI	Novem 16, 2
4	3	... a perverse twist on neorealism...	NaN	fresh	NaN	0	Cinema Scope	Octo 12, 2
...	...	...	...	...	...	...	...	
54427	2000	The real charm of this trifle is the deadpan c...	NaN	fresh	Laura Sinagra	1	Village Voice	Septem 24, 2
54428	2000	NaN	1/5	rotten	Michael Szymanski	0	Zap2it.com	Septem 21, 2
54429	2000	NaN	2/5	rotten	Emanuel Levy	0	EmanuelLevy.Com	July 2
54430	2000	NaN	2.5/5	rotten	Christopher Null	0	Filmcritic.com	Septem 7, 2
54431	2000	NaN	3/5	fresh	Nicolas Lacroix	0	Showbizz.net	Novem 12, 2

54432 rows × 8 columns

In [25]:
```
#Load 'tn.movie_budgets.csv'
mb = pd.read_csv(r"zippedData\tn.movie_budgets.csv.gz")
mb
```

Out[25]:

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

5782 rows × 6 columns

In [26]:
```
re = pd.read_csv(r"zippedData\tmdb.movies.csv.gz",index_col=0)
re
```

Out[26]:

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

26517 rows × 9 columns

In [4]:
```
#Here we want to unzip the zipped database file
zipped_db = r"zippedData\im.db.zip"
db_file = r"zippedData\im.db"

with zipfile.ZipFile(zipped_db, 'r') as zip_ref:
    if db_file in zip_ref.namelist():
        with zip_ref.open(db_file) as db:
            db_buffer = io.BytesIO(db_file.read())

mdb = sqlite3.connect(db_file).cursor()
mdb.execute("SELECT name FROM sqlite_master WHERE type = 'table';").fetchall()#This
```

Out[4]:
```
[('movie_basics',),
 ('directors',),
 ('known_for',),
 ('movie_akas',),
 ('movie_ratings',),
 ('persons',),
 ('principals',),
 ('writers',),
 ('existing_table',),
 ('mb',)]
```

In [5]:
```
conn=sqlite3.connect(db_file)
```

Now for the next section we want to get a closer look at all the tables from the database we have loaded above

In [6]:
```
mdb.execute(f"PRAGMA table_info('movie_basics');").fetchall()
```

Out[6]:
```
[(0, 'movie_id', 'TEXT', 0, None, 0),
 (1, 'primary_title', 'TEXT', 0, None, 0),
 (2, 'original_title', 'TEXT', 0, None, 0),
 (3, 'start_year', 'INTEGER', 0, None, 0),
 (4, 'runtime_minutes', 'REAL', 0, None, 0),
 (5, 'genres', 'TEXT', 0, None, 0)]
```

In [7]:
```
# To convert the movie_basics table to pandas dataframe
movie_basics=pd.read_sql(f"SELECT * FROM movie_basics;",conn)
```

In [24]:
```
mdb.execute(f"PRAGMA table_info('directors');").fetchall()
```

Out[24]:
```
[(0, 'movie_id', 'TEXT', 0, None, 0), (1, 'person_id', 'TEXT', 0, None, 0)]
```

In [9]:
```
# To convert the directors table to pandas dataframe
directors=pd.read_sql(f"SELECT * FROM directors;",conn)
```

In [10]:
```
mdb.execute(f"PRAGMA table_info('known_for');").fetchall()
```

Out[10]:
```
[(0, 'person_id', 'TEXT', 0, None, 0), (1, 'movie_id', 'TEXT', 0, None, 0)]
```

In [11]:
```
# To convert known_for table to pandas dataframe
known_for=pd.read_sql(f"SELECT * FROM known_for;",conn)
```

In [12]:
```
mdb.execute(f"PRAGMA table_info('movie_akas');").fetchall()
```

Out[12]:
```
[(0, 'movie_id', 'TEXT', 0, None, 0),
 (1, 'ordering', 'INTEGER', 0, None, 0),
 (2, 'title', 'TEXT', 0, None, 0),
 (3, 'region', 'TEXT', 0, None, 0),
 (4, 'language', 'TEXT', 0, None, 0),
 (5, 'types', 'TEXT', 0, None, 0),
 (6, 'attributes', 'TEXT', 0, None, 0),
 (7, 'is_original_title', 'REAL', 0, None, 0)]
```

In [13]:
```
# To convert movie_akas table to pandas dataframe
movie_akas=pd.read_sql(f"SELECT * FROM movie_akas;",conn)
```

In [14]:
```
mdb.execute(f"PRAGMA table_info('movie_ratings');").fetchall()
```

Out[14]:
```
[(0, 'movie_id', 'TEXT', 0, None, 0),
 (1, 'averagerating', 'REAL', 0, None, 0),
 (2, 'numvotes', 'INTEGER', 0, None, 0)]
```

In [15]:
```
# To convert movie_ratings table to pandas dataframe
movie_ratings=pd.read_sql(f"SELECT * FROM movie_ratings;",conn)
```

In [16]:
```
mdb.execute(f"PRAGMA table_info('persons');").fetchall()
```

Out[16]:
```
[(0, 'person_id', 'TEXT', 0, None, 0),
 (1, 'primary_name', 'TEXT', 0, None, 0),
 (2, 'birth_year', 'REAL', 0, None, 0),
 (3, 'death_year', 'REAL', 0, None, 0),
 (4, 'primary_profession', 'TEXT', 0, None, 0)]
```

In [17]:
```
# To convert persons table to pandas dataframe
persons=pd.read_sql(f"SELECT * FROM persons;",conn)
```

In [18]:
```
mdb.execute(f"PRAGMA table_info('principals');").fetchall()
```

Out[18]:
```
[(0, 'movie_id', 'TEXT', 0, None, 0),
 (1, 'ordering', 'INTEGER', 0, None, 0),
 (2, 'person_id', 'TEXT', 0, None, 0),
 (3, 'category', 'TEXT', 0, None, 0),
 (4, 'job', 'TEXT', 0, None, 0),
 (5, 'characters', 'TEXT', 0, None, 0)]
```

In [19]:
```
# To convert principals table to pandas dataframe
principals=pd.read_sql(f"SELECT * FROM principals;",conn)
```

In [20]:
```
mdb.execute(f"PRAGMA table_info('writers');").fetchall()
```

Out[20]:
```
[(0, 'movie_id', 'TEXT', 0, None, 0), (1, 'person_id', 'TEXT', 0, None, 0)]
```

In [22]:
```
# To convert writers table to pandas dataframe
writers=pd.read_sql(f"SELECT * FROM writers;",conn)
```

This will now make it easier to find the columns that we would be using to connect two or more of the tables together. Now that we've seen what they contain and the data types of the columns. Now the task ahead will be to try and find a way of matching the database extracted to `mdb` with the other files in order to be able to analyse the data and come out insights that could help the company make the decision on what kind of movie studio they want to open.

So the best way to look at the kind of data that would be needed would be trying to answer the following questions

1. What is the total profit made by movies depending on the genres? and which genres have the most profit
2. What is the most popular movie genre among customers?
3. when is the most appropriate time to release a movie in order to make the most amount of profit?

1. What is the total profit made by movies depending on genres? And which genres have the most profit

For this question we willl want to create a column in the movies budget called `Gross Profit`. This column wil be consisting of total gross profit = domestic gross + international gross and then we will use that get the gross profit which will be calculated by taking total gros profit and subtracting from production budget. This will help us know what if there was actually a true profit made.

In [27]:
```
#we know that the data here are in type object so we start by converting the requir
mb['production_budget'] = pd.to_numeric(mb['production_budget'].replace({r'\$': '',
mb['domestic_gross'] = pd.to_numeric(mb['domestic_gross'].replace({r'\$': '', r',':
mb['worldwide_gross'] = pd.to_numeric(mb['worldwide_gross'].replace({r'\$': '', r',
```

In [28]:
```
#verifying that they have converted into type int
mb.dtypes
```

Out[28]:
```
id                    int64
release_date         object
movie                object
production_budget     int64
domestic_gross        int64
worldwide_gross       int64
dtype: object
```

In [29]:
```
# Now let's create a new column 'gross_profit and input the relevant data into it
mb['grossprofit'] = (mb['domestic_gross'] + mb['worldwide_gross']) - mb['production
mb
```

Out[29]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross	g
0	1	Dec 18, 2009	Avatar	425000000	760507625	2776345279	3
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	149762350	-
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	1403013963	1
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1316721747	1
...	...	...	...	...	...	...	...
5777	78	Dec 31, 2018	Red 11	7000	0	0	
5778	79	Apr 2, 1999	Following	6000	48482	240495	
5779	80	Jul 13, 2005	Return to the Land of Wonders	5000	1338	1338	
5780	81	Sep 29, 2015	A Plague So Pleasant	1400	0	0	
5781	82	Aug 5, 2005	My Date With Drew	1100	181041	181041	

5782 rows × 7 columns

In [275…

```
#Here we are just verifying to ensure that there is no missing data in the dataset
mb.isnull().sum()
```

Out[275…

```
id                 0
release_date       0
movie              0
production_budget  0
domestic_gross     0
worldwide_gross    0
grossprofit        0
dtype: int64
```

Now that we have created a new column 'gross_profit we can visibly see that there have been movies with losses and movies with profit. The next step is now to find a way of getting the genres of each and every movie so as to make grouping them into genres a lot easier. The table `movie_basics` in the database will be the best way seeing as it has the genres labled out. So we will want to join the `movie_basics` table with the `mb` dataframe created above

In [276…

```
movie_basics.head(15)
```

Out[276...

	movie_id	primary_title	original_title	start_year	runtime_minutes	
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crim
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biograph
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	Comec
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama
5	tt0111414	A Thin Life	A Thin Life	2018	75.0	
6	tt0112502	Bigfoot	Bigfoot	2017	NaN	Horro
7	tt0137204	Joe Finds Grace	Joe Finds Grace	2017	83.0	Adventure,Animation
8	tt0139613	O Silêncio	O Silêncio	2012	NaN	Documentar
9	tt0144449	Nema aviona za Zagreb	Nema aviona za Zagreb	2012	82.0	B
10	tt0146592	Pál Adrienn	Pál Adrienn	2010	136.0	
11	tt0154039	So Much for Justice!	Oda az igazság	2010	100.0	
12	tt0159369	Cooper and Hemingway: The True Gen	Cooper and Hemingway: The True Gen	2013	180.0	Docu
13	tt0162942	Children of the Green Dragon	A zöld sárkány gyermekei	2010	89.0	
14	tt0170651	T.G.M. - osvoboditel	T.G.M. - osvoboditel	2018	60.0	Docu

In [277...

```
movie_basics['genres'] = movie_basics['genres'].apply(lambda x: x.split(',') if isi
movie_basics_expanded = movie_basics.explode('genres')
movie_basics_expanded['genres'] = movie_basics_expanded['genres'].str.strip().str.l
movie_basics_expanded = movie_basics_expanded.reset_index(drop=True)
movie_basics_expanded.head(15)
```

Out[277…

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	action
1	tt0063540	Sunghursh	Sunghursh	2013	175.0	crime
2	tt0063540	Sunghursh	Sunghursh	2013	175.0	drama
3	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	biography
4	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	drama
5	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	drama
6	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	comedy
7	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	drama
8	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	comedy
9	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	drama
10	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	fantasy
11	tt0111414	A Thin Life	A Thin Life	2018	75.0	comedy
12	tt0112502	Bigfoot	Bigfoot	2017	NaN	horror
13	tt0112502	Bigfoot	Bigfoot	2017	NaN	thriller
14	tt0137204	Joe Finds Grace	Joe Finds Grace	2017	83.0	adventure

In [278…

```
# Rename the column movie to title
mb = mb.rename(columns={'movie':'title'})
mb
```

Out[278...

	id	release_date	title	production_budget	domestic_gross	worldwide_gross	g
0	1	Dec 18, 2009	Avatar	425000000	760507625	2776345279	3
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	149762350	-
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	1403013963	1
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1316721747	1
...	...	...	...	...	...	...	
5777	78	Dec 31, 2018	Red 11	7000	0	0	
5778	79	Apr 2, 1999	Following	6000	48482	240495	
5779	80	Jul 13, 2005	Return to the Land of Wonders	5000	1338	1338	
5780	81	Sep 29, 2015	A Plague So Pleasant	1400	0	0	
5781	82	Aug 5, 2005	My Date With Drew	1100	181041	181041	

5782 rows × 7 columns

In [279...

```
mbd = pd.merge(mb,re, on='title', how='inner')
mbd
```

Out[279…

	id_x	release_date_x	title	production_budget	domestic_gross	worldwide_gro
0	1	Dec 18, 2009	Avatar	425000000	760507625	27763452
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	10456638
2	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	14030139
3	7	Apr 27, 2018	Avengers: Infinity War	300000000	678815482	20481342
4	9	Nov 17, 2017	Justice League	300000000	229024295	6559452
…	…	…	…	…	…	
2380	49	Sep 1, 2015	Exeter	25000	0	4897
2381	51	Apr 21, 2015	Ten	25000	0	
2382	54	Dec 31, 2014	Dry Spell	22000	0	
2383	56	Jan 4, 2013	All Superheroes Must Die	20000	0	
2384	73	Jan 13, 2012	Newlyweds	9000	4584	45

2385 rows × 15 columns

In [280…

```
mbd = mbd.drop(columns=['release_date_x','original_language', 'original_title', 're
mbd.head(15)
```

Out[280…

	title	production_budget	domestic_gross	worldwide_gross	grossprofit	popularit
0	Avatar	425000000	760507625	2776345279	3111852904	26.52
1	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	876127750	30.57
2	Avengers: Age of Ultron	330600000	459005868	1403013963	1531419831	44.38
3	Avengers: Infinity War	300000000	678815482	2048134200	2426949682	80.77
4	Justice League	300000000	229024295	655945209	584969504	34.95
5	Justice League	300000000	229024295	655945209	584969504	34.95
6	Spectre	300000000	200074175	879620923	779695098	30.31
7	Spectre	300000000	200074175	879620923	779695098	30.31
8	The Dark Knight Rises	275000000	448139099	1084439099	1257578198	26.22
9	Solo: A Star Wars Story	275000000	213767512	393151347	331918859	29.50
10	The Lone Ranger	275000000	89302115	260002115	74304230	12.48
11	John Carter	275000000	73058679	282778100	80836779	18.54
12	Tangled	260000000	200821936	586477240	527299176	21.51
13	Captain America: Civil War	250000000	408084349	1140069413	1298153762	39.13
14	Batman v Superman: Dawn of Justice	250000000	330360194	867500281	947860475	28.08

In [281…

```
movie_basics_expanded = movie_basics_expanded.rename(columns={'primary_title':'titl
movie_total = pd.merge(mbd, movie_basics_expanded, on = 'title', how='inner')
```

```
movie_total = movie_total.drop(columns=['production_budget', 'domestic_gross','worl
movie_total.head(15)
```

Out[281...

	title	grossprofit	popularity	vote_average	vote_count	original_title	start_year
0	Avatar	3111852904	26.526	7.4	18676	Abatâ	2011
1	Pirates of the Caribbean: On Stranger Tides	876127750	30.579	6.4	8571	Pirates of the Caribbean: On Stranger Tides	2011
2	Pirates of the Caribbean: On Stranger Tides	876127750	30.579	6.4	8571	Pirates of the Caribbean: On Stranger Tides	2011
3	Pirates of the Caribbean: On Stranger Tides	876127750	30.579	6.4	8571	Pirates of the Caribbean: On Stranger Tides	2011
4	Avengers: Age of Ultron	1531419831	44.383	7.3	13457	Avengers: Age of Ultron	2015
5	Avengers: Age of Ultron	1531419831	44.383	7.3	13457	Avengers: Age of Ultron	2015
6	Avengers: Age of Ultron	1531419831	44.383	7.3	13457	Avengers: Age of Ultron	2015
7	Avengers: Infinity War	2426949682	80.773	8.3	13948	Avengers: Infinity War	2018
8	Avengers: Infinity War	2426949682	80.773	8.3	13948	Avengers: Infinity War	2018
9	Avengers: Infinity War	2426949682	80.773	8.3	13948	Avengers: Infinity War	2018
10	Justice League	584969504	34.953	6.2	7510	Justice League	2017
11	Justice League	584969504	34.953	6.2	7510	Justice League	2017
12	Justice League	584969504	34.953	6.2	7510	Justice League	2017

	title	grossprofit	popularity	vote_average	vote_count	original_title	start_year
13	Justice League	584969504	34.953	6.2	7510	Justice League	2017
14	Justice League	584969504	34.953	6.2	7510	Justice League	2017

In [323…

```
profit_by_genres = movie_total.groupby('genres').agg(total_grossprofit=pd.NamedAgg(
profit_by_genres
```

Out[323...

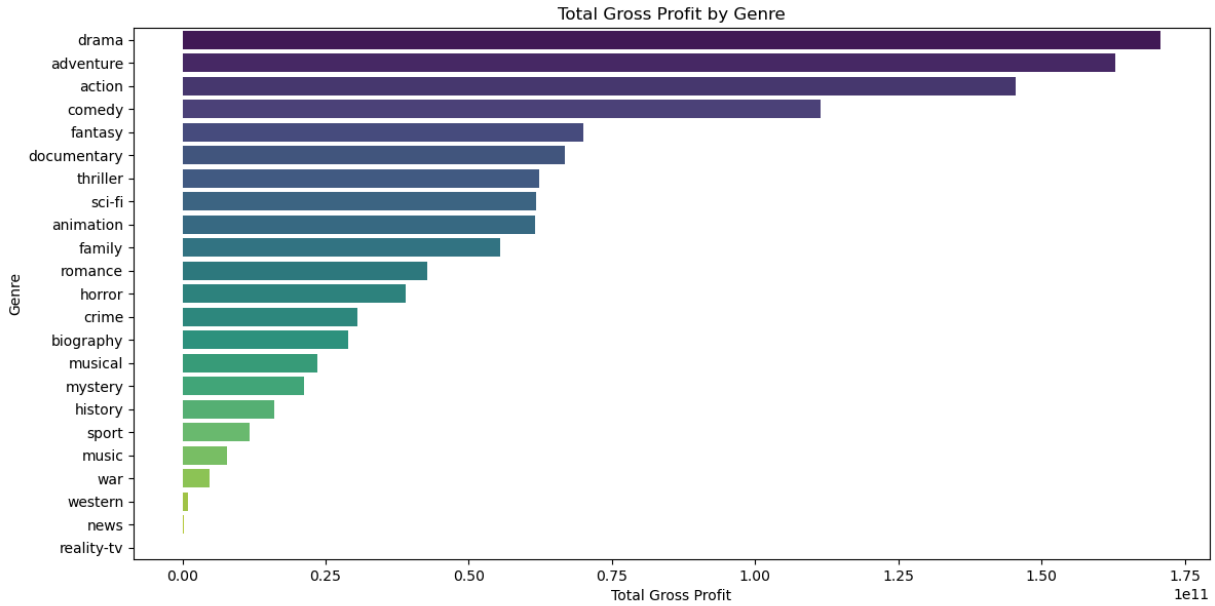
genres	total_grossprofit	avg_popularity	avg_vote_average	total_vote_count
drama	170846486340	7.318953	6.275980	2150496
adventure	162822821568	15.095244	6.225709	1687609
action	145416892272	12.811314	5.985071	1804730
comedy	111487058098	9.517845	6.132127	1127926
fantasy	70092990068	13.945504	6.035776	631950
documentary	66693542526	4.741006	6.055334	426962
thriller	62346116942	7.963337	5.918982	766995
sci-fi	61673474274	13.994959	6.111934	820414
animation	61636527866	10.666538	6.232487	392193
family	55466012392	7.988199	6.110487	352345
romance	42702041389	7.358341	6.332294	439793
horror	38952230024	7.702130	5.698798	503505
crime	30612965195	9.799904	6.090359	546864
biography	28998366460	8.716848	6.489634	388702
musical	23526646528	12.798750	6.461364	125873
mystery	21184691452	8.935328	6.052453	371325
history	16107064978	7.682735	6.382119	156368
sport	11717687285	6.454722	6.357732	85135
music	7742673970	8.052589	6.424211	93937
war	4789180780	4.409105	6.122368	34251
western	980105427	7.164920	5.916000	29693
news	180628377	9.405200	6.140000	5996
reality-tv	-2000000	6.397000	6.400000	2148

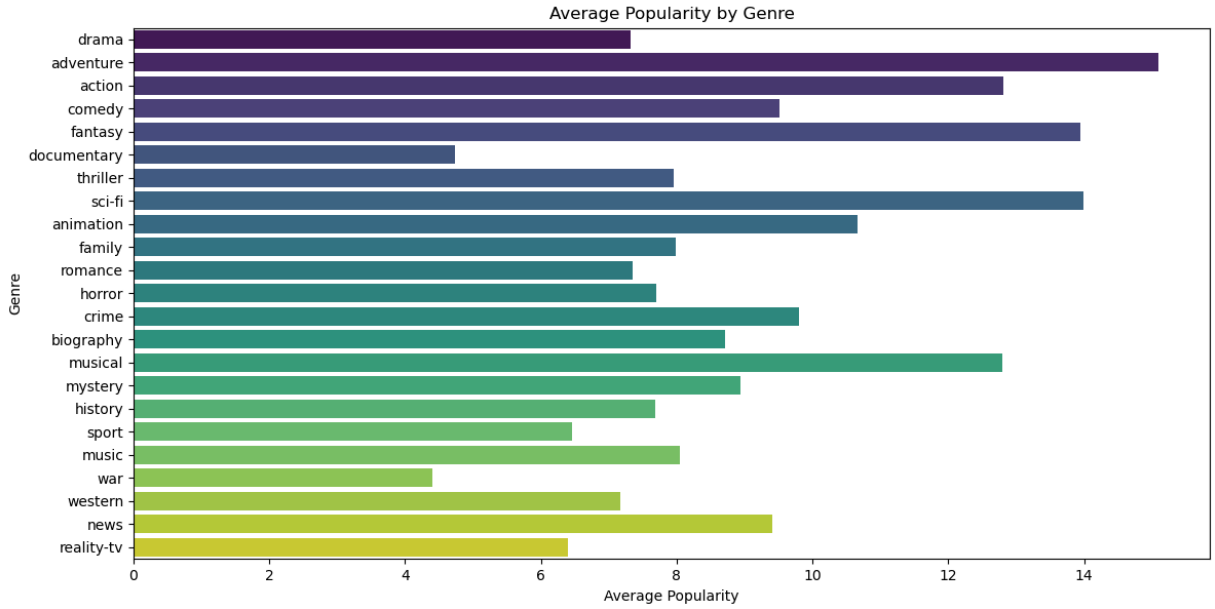
Now that we have the data that we need, it is time for us to find a way in which we can view this data. So the best way will be to draw a graph that will give us a view of what is going on. So in the cell below we'll create the code to generate the neccessary graphs

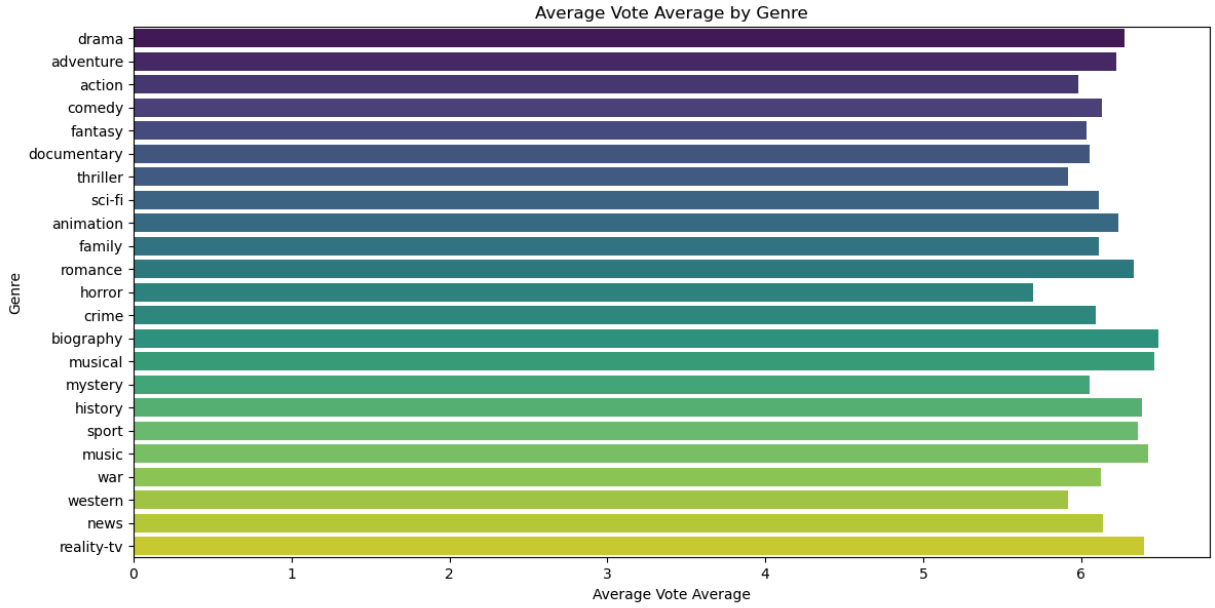
In [324...

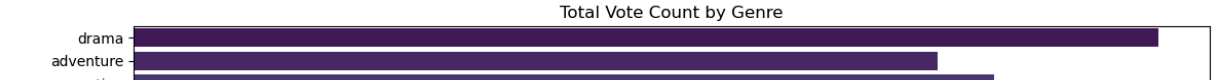
```
fig, axes = plt.subplots(4, 1, figsize=(12, 24))
# Total gross profit by genre
sns.barplot(x=profit_by_genres['total_grossprofit'], y=profit_by_genres.index, ax=a
axes[0].set_title('Total Gross Profit by Genre')
axes[0].set_xlabel('Total Gross Profit')
```

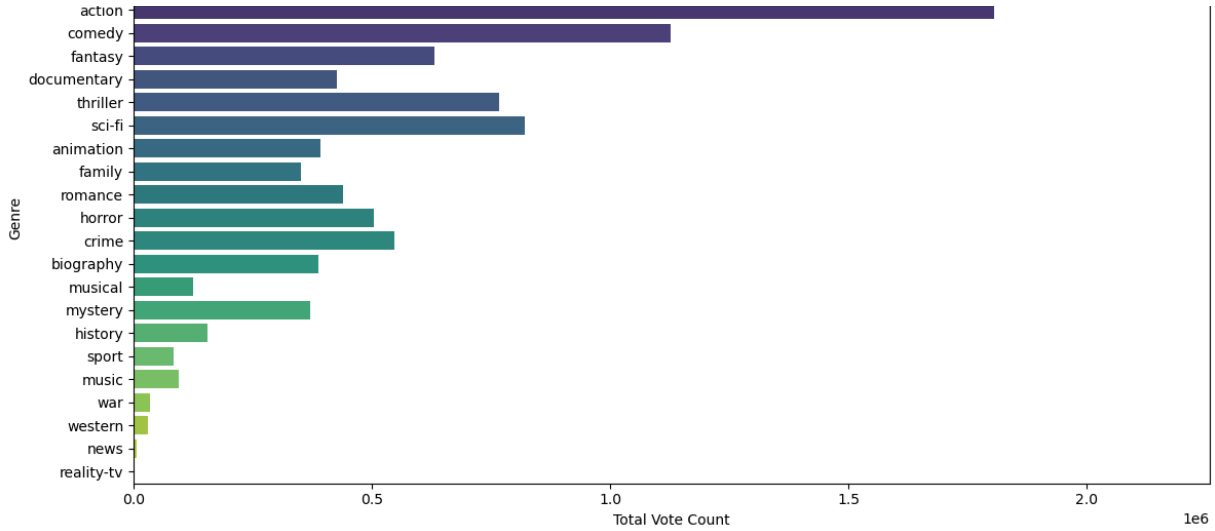
```
axes[0].set_ylabel('Genre')

# Average popularity by genre
sns.barplot(x=profit_by_genres['avg_popularity'], y=profit_by_genres.index, ax=axes
axes[1].set_title('Average Popularity by Genre')
axes[1].set_xlabel('Average Popularity')
axes[1].set_ylabel('Genre')

# Average vote average by genre
sns.barplot(x=profit_by_genres['avg_vote_average'], y=profit_by_genres.index, ax=ax
axes[2].set_title('Average Vote Average by Genre')
axes[2].set_xlabel('Average Vote Average')
axes[2].set_ylabel('Genre')

# Total vote count by genre
sns.barplot(x=profit_by_genres['total_vote_count'], y=profit_by_genres.index, ax=ax
axes[3].set_title('Total Vote Count by Genre')
axes[3].set_xlabel('Total Vote Count')
axes[3].set_ylabel('Genre')

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


Total Gross Profit by Genre
drama
adventure
action
comedy
fantasy
documentary
thriller
sci-fi
animation
family
romance
horror
crime
biography
musical
mystery
history
sport
music
war
western
news
reality-tv
Genre
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e11
Total Gross Profit


Average Popularity by Genre
drama
adventure
action
comedy
fantasy
documentary
thriller
sci-fi
animation
family
romance
horror
crime
biography
musical
mystery
history
sport
music
war
western
news
reality-tv
Genre
0
2
4
6
8
10
12
14
Average Popularity


Average Vote Average by Genre
drama
adventure
action
comedy
fantasy
documentary
thriller
sci-fi
animation
family
romance
horror
crime
biography
musical
mystery
history
sport
music
war
western
news
reality-tv
Genre
0
1
2
3
4
5
6
Average Vote Average


Total Vote Count by Genre
drama
adventure



Pay attention to the fact that the graphs are all aranged in accordance with the highest total_grossprofit as away of showing the data in a uniform manner. Now from the graphs above we can come to the following conlcusions:

1. There is no direct relation to the genre with the highest amount of profit and its correlation.
2. There is not much difference between the average ratings accross all the genres
3. The vote count of a genre can also be related to its gross profit as it is seen that the genres with high profits also had a large vote count and vice versa

2. What is the most popular movie genre among customers?

Now in order to answer this we would want to look at the ratings that are given to the movies and which genres have the genral highest ratings. The best way to get this info would be to find a way to merge the movie_basics table and the movie ratings table.

In [283…
```
#A quick look at movie_basics_expanded to see what the table holds
movie_basics_expanded.head()
```

Out[283…

	movie_id	title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	action
1	tt0063540	Sunghursh	Sunghursh	2013	175.0	crime
2	tt0063540	Sunghursh	Sunghursh	2013	175.0	drama
3	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	biography
4	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	drama

In [284…
```
#This is a view of the table that has the movie ratings
movie_ratings.head()
```

Out[284…

	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 [285…
```
#check if there are any missing values
movie_basics_expanded.isna().sum()
```

Out[285…
```
movie_id              0
title                 0
original_title       21
start_year            0
runtime_minutes    39054
genres              5408
dtype: int64
```

In [286…
```
# dropping the neccessary columns
movie_basics_expanded.dropna(subset=['original_title','runtime_minutes','genres'],i
```

In [287…
```
#check for missing values after droping the columns in movie_basics_expanded
movie_basics_expanded.isna().sum()
```

Out[287…
```
movie_id          0
title             0
original_title    0
start_year        0
runtime_minutes   0
genres            0
dtype: int64
```

In [288…
```
#Cheeck for missing values in movie_ratings
movie_ratings.isna().sum()
```

Out[288…
```
movie_id        0
averagerating   0
numvotes        0
dtype: int64
```

In [289…
```
#combining the movie_basics_expanded with the movie_ratings to have one cleat dataf
mbr=pd.merge(movie_basics_expanded,movie_ratings, on='movie_id', how='inner')
mbr.head(15)
```

Out[289…

	movie_id	title	original_title	start_year	runtime_minutes	genres	averag
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	action	
1	tt0063540	Sunghursh	Sunghursh	2013	175.0	crime	
2	tt0063540	Sunghursh	Sunghursh	2013	175.0	drama	
3	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	biography	
4	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	drama	
5	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	drama	
6	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	comedy	
7	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	drama	
8	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	fantasy	
9	tt0137204	Joe Finds Grace	Joe Finds Grace	2017	83.0	adventure	
10	tt0137204	Joe Finds Grace	Joe Finds Grace	2017	83.0	animation	
11	tt0137204	Joe Finds Grace	Joe Finds Grace	2017	83.0	comedy	
12	tt0146592	Pál Adrienn	Pál Adrienn	2010	136.0	drama	
13	tt0154039	So Much for Justice!	Oda az igazság	2010	100.0	history	
14	tt0159369	Cooper and Hemingway: The True Gen	Cooper and Hemingway: The True Gen	2013	180.0	documentary	

In [291…

```
#check for any duplicate values
mbr.duplicated().sum()
```

Out[291… 0

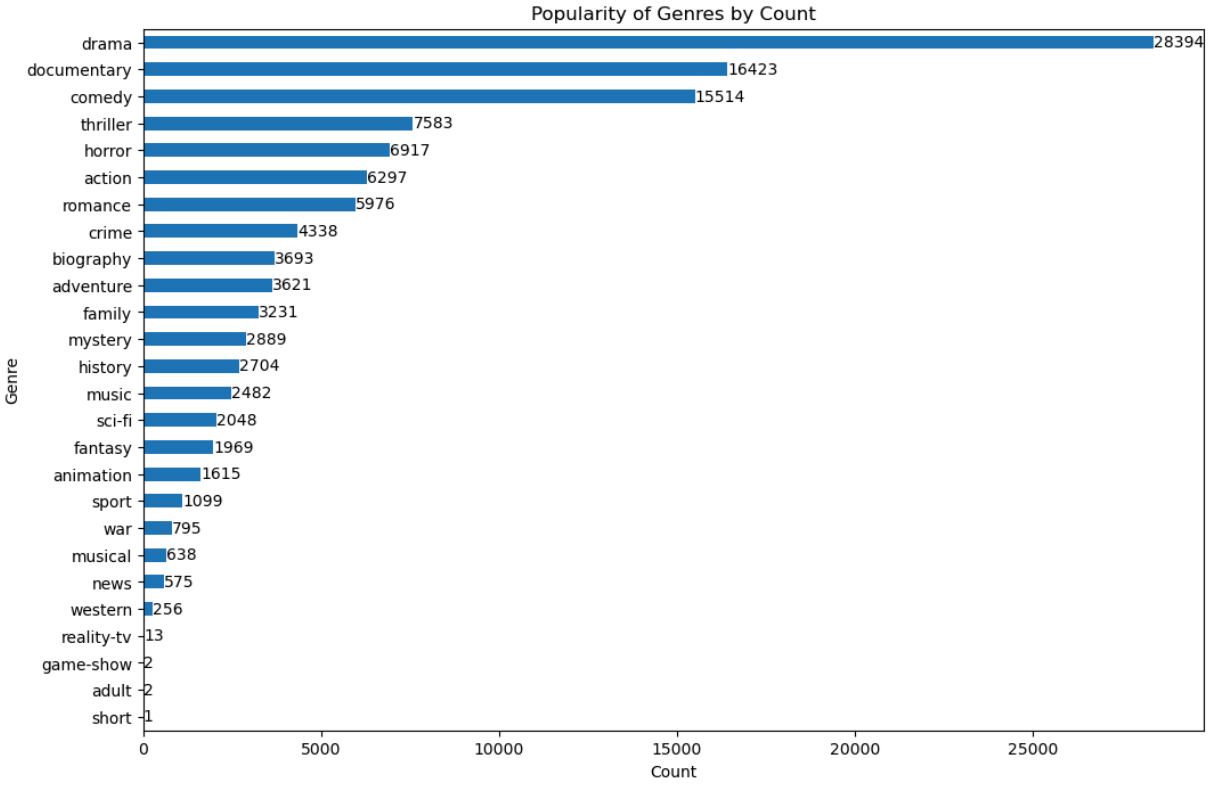
In [294…
```
mbr_unique_genres= mbr['genres'].unique()
for genre in mbr_unique_genres:
    mbr[genre] = mbr['genres'].apply(lambda x: 1 if genre in x else 0)

mbr.head(15)
```

Out[294...

	movie_id	title	original_title	start_year	runtime_minutes	genres	averag
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	action	
1	tt0063540	Sunghursh	Sunghursh	2013	175.0	crime	
2	tt0063540	Sunghursh	Sunghursh	2013	175.0	drama	
3	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	biography	
4	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	drama	
5	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	drama	
6	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	comedy	
7	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	drama	
8	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	fantasy	
9	tt0137204	Joe Finds Grace	Joe Finds Grace	2017	83.0	adventure	
10	tt0137204	Joe Finds Grace	Joe Finds Grace	2017	83.0	animation	
11	tt0137204	Joe Finds Grace	Joe Finds Grace	2017	83.0	comedy	
12	tt0146592	Pál Adrienn	Pál Adrienn	2010	136.0	drama	
13	tt0154039	So Much for Justice!	Oda az igazság	2010	100.0	history	
14	tt0159369	Cooper and Hemingway: The True Gen	Cooper and Hemingway: The True Gen	2013	180.0	documentary	

15 rows × 34 columns

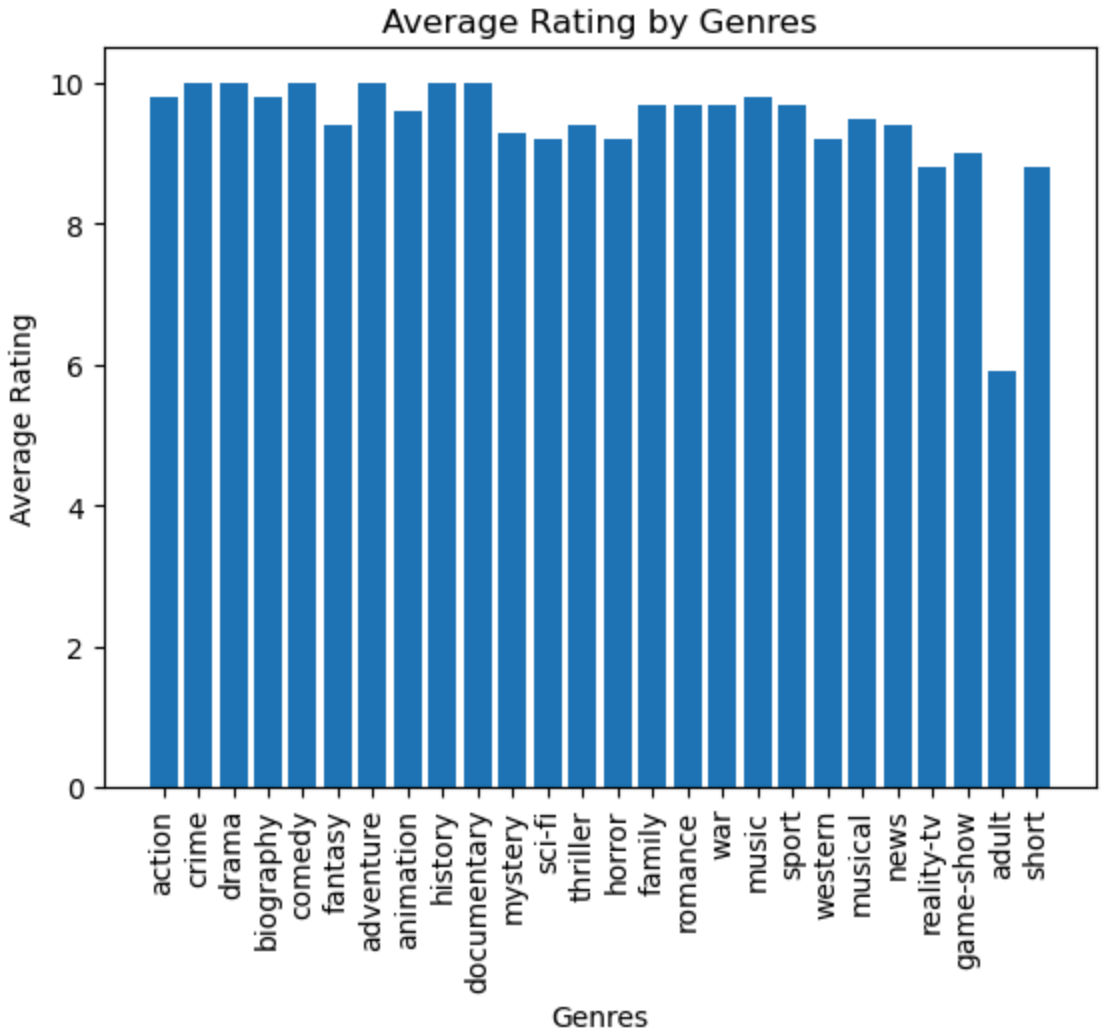
In [296…
```
#Now let us crate a bargrph that shows all this information and sort with the most
genre_counts = mbr[mbr_unique_genres].sum()
plt.figure(figsize=(12, 8))
ax = genre_counts.sort_values().plot(kind='barh')
for p in ax.patches:
    width = p.get_width()
    ax.text(width + 0.1, p.get_y() + p.get_height() / 2, int(width),
            ha='left', va='center', color='black')

plt.title('Popularity of Genres by Count')
plt.xlabel('Count')
plt.ylabel('Genre')
plt.show();
```



From the graph above we can note the following:

- There is no other genre more popular than drama due to its large following
- The top 5 genres with a reasonable popularity count are : Drama, Documentary, Comedy, Thriller and horror while only the first three have a popularity count of over 15k

In [298…
```
#Here we are going to plot a simple graph of the movie genres and their average rat
plt.bar(mbr['genres'], mbr['averagerating'])
plt.xlabel('Genres')
plt.ylabel('Average Rating')
plt.title('Average Rating by Genres')
plt.xticks(rotation=90)
plt.show()
```



3.When is the most appropriate time to release a movie in order to make the most amount of profit?

In order to get this, we'll have to look at the movie budgets table.

In [300…

```
mb.head()
```

Out[300…

	id	release_date	title	production_budget	domestic_gross	worldwide_gross	gross
0	1	Dec 18, 2009	Avatar	425000000	760507625	2776345279	31118
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	410600000	241063875	1045663875	8761
2	3	Jun 7, 2019	Dark Phoenix	350000000	42762350	149762350	-1574
3	4	May 1, 2015	Avengers: Age of Ultron	330600000	459005868	1403013963	15314
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	317000000	620181382	1316721747	16199

In [301…

```
# converting relevant rows from object to integer data type
mb[mb.columns[3:6]]= mb[mb.columns[3:6]].replace('[\$,]', '' ,regex=True).astype(fl
```

```
<>:2: SyntaxWarning: invalid escape sequence '\$'
<>:2: SyntaxWarning: invalid escape sequence '\$'
C:\Users\User\AppData\Local\Temp\ipykernel_19388\4004669862.py:2: SyntaxWarning: inv
alid escape sequence '\$'
  mb[mb.columns[3:6]]= mb[mb.columns[3:6]].replace('[\$,]', '' ,regex=True).astype(f
loat)
```

In [302…

```
#changing the release_date column to date_time data type.
mb["release_date"]=mb["release_date"].astype('datetime64[ns]')
```

Here, we create a new column 'wrldwide_profit' that gives the worldwide profits of each movie.

In [303…

```
# Creating a new column 'worldwide_profit' to display profits.
mb['worldwide_profit']=mb['worldwide_gross']-mb['production_budget']
```

In [304…

```
mb.head()
```

Out[304…

	id	release_date	title	production_budget	domestic_gross	worldwide_gross	gross
0	1	2009-12-18	Avatar	425000000.0	760507625.0	2.776345e+09	31118
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09	8761
2	3	2019-06-07	Dark Phoenix	350000000.0	42762350.0	1.497624e+08	-1574
3	4	2015-05-01	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09	15314
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000.0	620181382.0	1.316722e+09	16199

In [305…

```
# Creating a column of the months only
mb["month"]=mb['release_date'].dt.month
```

In [306…

```
mb.head()
```

Out[306…

	id	release_date	title	production_budget	domestic_gross	worldwide_gross	gross
0	1	2009-12-18	Avatar	425000000.0	760507625.0	2.776345e+09	31118
1	2	2011-05-20	Pirates of the Caribbean: On Stranger Tides	410600000.0	241063875.0	1.045664e+09	8761
2	3	2019-06-07	Dark Phoenix	350000000.0	42762350.0	1.497624e+08	-1574
3	4	2015-05-01	Avengers: Age of Ultron	330600000.0	459005868.0	1.403014e+09	15314
4	5	2017-12-15	Star Wars Ep. VIII: The Last Jedi	317000000.0	620181382.0	1.316722e+09	16199

we then group by months to find the average profits of each month over the years.

In [309…
```
# Grouping the profits by months and displaying the average gross profits of each m
profits=mb.groupby('month')['worldwide_profit'].mean()
profits
```

Out[309…
```
month
1     2.572033e+07
2     4.349811e+07
3     4.985129e+07
4     3.611743e+07
5     1.151328e+08
6     9.942391e+07
7     9.841746e+07
8     3.542232e+07
9     2.488078e+07
10    2.907190e+07
11    9.314157e+07
12    6.844157e+07
Name: worldwide_profit, dtype: float64
```

In [310…
```
# plotting a line-graph of average profits of each month over the years

months=["Jan","Feb","Mar","april","May","June","July","Aug","Sep","Oct","Nov","Dec"
fig,ax=plt.subplots()
ax.plot(months,profits
       )
ax.set_title("Average monthly profits over the years")
ax.set_xlabel("release_Month")
ax.set_ylabel("Gross profits in '10 millions' ");
```

From the above graph we can deduce that;

- There is high growth in interest in between the month of April and May and the same for October to November
- Movies released in the months May,June,July and November have yielded the most gross profits.
- Movies released in January,April,Aug,September and October have yielded the least profits.

CONCLUSIONS AND RECOMENDATIONS

Conclusions

So here are the questions we sought to answer and there conclusions from the analysis above:

1. What is the total profit made by movies depending on the genres?And which genres have the most profit? For this question we will look at the top 5 genres in our list:
    - drama
    - adventure
    - action
    - comedy
    - fantasy Based on our findings these are the most profitable genres to start with given that the company will be able to match or excede the production budget of movies that fall in these genres.
2. What is the most popular movie genre among customers? Here we considered the vote count of the genres and their average ratings across all movies found in a specific genre. So for the vote count of the genres it wa a clear indication that the drama genre had the most amount of fans seeing as it clear above the rest and then it wasfollowed by documentary and comedy genres. As for the average rating it was easy to spot that the genres were all roughly having the same range in the rating s and the difference was not so significant

3. when is the most appropriate time to release a movie in order to make the most amount of profit? Now after grouping the data is was clear that there was two peak seasons in when the amount of profit increased dramatically . The first was from the begining of April towards May which then saw that over June and Jully the amount ofprofit was fairly high . The the second peak period was from October to November but unfortunately it was evidence of the decline in profit from November to December. So from that analysis the best time to release a movie would be in the months of April and November

Recomendations

1. Seeing as to how the company will be venturing into the movie business as a beginner it would be advised to start with a middle tier movie genre as a way to build raport among customers and to build the name of the company's studio. A good genre to venturee into would be peobably `thriller`, `Sci-Fi` and `animation` primarily because they are in the middle range of the genres in ternms of popularity and also the profit it brings in. Not the most but defenitely ot is substansial for a begining studio

2. if there is going to be a release date for any movie made it should be in the months of April and October as thos have beeen seen to see a rise in profits which can mainly be due to the customers have a lot of free time in those months

This project was prepared and presented by:

1. Baraka Nyamai
2. Maureen Imanene
3. Grace Kimotho
4. Sharon Mungai
5. Alvin Asingo