Project: Movie Data Analysis

Table of Contents

Introduction

This is a movie dataset which i got from kaggle. This Dataset contains contains information about 10,000 movies which was collected from The Movie Database (TMDb). This Dataset contains numerous information about movies within the year span of 1960-2016 across all genres.

The analysis i am going to conduct on this Dataset will answer the following questions:

1. Which genres are the most popular from year to year?
2. Does the popularity of a movie have any effect on the revenue the movie generates
3. Which genre of movie produces the highest revenue?
4. Does the average vote collated for each genre have a significant effect on the revenue?

The first step is to import all the packages that will be needed for this analysis. This is important as without them some of the codes written will produce an error message when you try run it

In [1]:
```
# This is to import all the packages that will needed during my analysis
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# Remember to include a 'magic word' so that your visualizations are plotted
#   inline with the notebook. See this page for more:
#   http://ipython.readthedocs.io/en/stable/interactive/magics.html
```

Data Wrangling

This is the first section of the analysis. To begin i will load in my data from the folder where it is stored, check for cleanliness, trim and clean my dataset for the analysis stage

General Properties

In [3]:
```
# This is to load in my data
df= pd.read_csv('tmdb-movies.csv')
#I didn't use the sep function because the data is already seperated by comma
```

After reading in my data, i am going to check for the informations about the dataframe and if there are any irregularities

In [4]:
```
#To know how many rows and columns i have in my dataframe
df.shape
```

Out[4]:
```
(10866, 21)
```

The above output indicates that there are 10,866 rows and 21 columns in the dataframe. The '.shape' function provides a summary on the structure of the dataframe

In [5]:
```
#To see the design of the dataframe
df.head()
```

Out[5]:

	id	imdb_id	popularity	budget	revenue	original_title	cast	
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.tl
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	http://.
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	

5 rows × 21 columns

The '.head()' function displays the first 5 rows in the dataframe and the column labels. Looking at the labels it is okay as it is for my analysis so i will not be changing any labels.

In [6]:
```
#To know the datatype associated with each column
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    10866 non-null  int64  
 1   imdb_id               10856 non-null  object 
 2   popularity            10866 non-null  float64
 3   budget                10866 non-null  int64  
 4   revenue               10866 non-null  int64  
 5   original_title        10866 non-null  object 
 6   cast                  10790 non-null  object 
 7   homepage              2936 non-null   object 
 8   director              10822 non-null  object 
 9   tagline               8042 non-null   object 
 10  keywords              9373 non-null   object 
 11  overview              10862 non-null  object 
 12  runtime               10866 non-null  int64  
 13  genres                10843 non-null  object 
 14  production_companies  9836 non-null   object 
 15  release_date          10866 non-null  object 
 16  vote_count            10866 non-null  int64  
 17  vote_average          10866 non-null  float64
 18  release_year          10866 non-null  int64  
 19  budget_adj            10866 non-null  float64
 20  revenue_adj           10866 non-null  float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

The '.info()' function displays the makeup of each column. It show the datatype, total number of cells with values for each column. The result displayed shows that there are some columns with null values of which i will have to clean to ensure the dataframe is ready for analysis

In [7]:
```
#To show the mean and other statistics about numerical columns
df.describe()
```

Out[7]:

	id	popularity	budget	revenue	runtime	vote_count	vote_av
count	10866.000000	10866.000000	1.086600e+04	1.086600e+04	10866.000000	10866.000000	10866.0
mean	66064.177434	0.646441	1.462570e+07	3.982332e+07	102.070863	217.389748	5.9
std	92130.136561	1.000185	3.091321e+07	1.170035e+08	31.381405	575.619058	0.9
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.000000	1.5
25%	10596.250000	0.207583	0.000000e+00	0.000000e+00	90.000000	17.000000	5.4
50%	20669.000000	0.383856	0.000000e+00	0.000000e+00	99.000000	38.000000	6.0
75%	75610.000000	0.713817	1.500000e+07	2.400000e+07	111.000000	145.750000	6.6
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	9.2

The above shows statistical values of columns that have numerical data. It will not show any of the columns that are non-numerical

Looking at the structures of the dataframe above, it shows that there are columns and row that will be dropped as they have null values and will not be relevant in the analysis of the data

After discussing the structure of the data and any problems that need to be cleaned, perform those cleaning steps in the second part of this section.

In [8]:
```
#To remove some of the columns i will not need for my analysis
df.drop(['id', 'imdb_id', 'homepage', 'release_date','tagline','keywords','production_
df.info()
# This will display the information about the new columns left in the dataframe
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   popularity      10866 non-null  float64
 1   budget          10866 non-null  int64  
 2   revenue         10866 non-null  int64  
 3   original_title  10866 non-null  object 
 4   cast            10790 non-null  object 
 5   director        10822 non-null  object 
 6   overview        10862 non-null  object 
 7   runtime         10866 non-null  int64  
 8   genres          10843 non-null  object 
 9   vote_count      10866 non-null  int64  
 10  vote_average    10866 non-null  float64
 11  release_year    10866 non-null  int64  
 12  budget_adj      10866 non-null  float64
 13  revenue_adj     10866 non-null  float64
dtypes: float64(4), int64(5), object(5)
memory usage: 1.2+ MB
```

The above code removed 6 of the columns i didnt need as they dont have any correlation to the questions i want to use this dataset to answer. As the cleaning continues there will be other rows and columns that i will drop

The next step is to check for duplicates and drop them

In [9]:
```
# This is to know how many rows are duplicated
sum(df.duplicated())
```

Out[9]:
```
1
```

In [10]:
```
# To drop all duplicates
df.drop_duplicates(inplace=True)
```

In [11]:
```
# This is to know the current amount of rows and colums we have after dropping
df.shape
```

Out[11]:
```
(10865, 14)
```

In [12]:
```
# To know the amount of null values per column
df.isna().sum()
```

Out[12]:
```
popularity         0
budget             0
revenue            0
original_title     0
cast              76
director          44
overview           4
runtime            0
genres            23
vote_count         0
vote_average       0
release_year       0
budget_adj         0
revenue_adj        0
dtype: int64
```

I am going to filter out the rows without null values because the rows with the null values are not useful right now for my analysis but can be useful later on

In [13]:
```
#rows without null values
filtered_df = df.loc[df.notna().all(1)]
filtered_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10729 entries, 0 to 10865
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   popularity      10729 non-null  float64
 1   budget          10729 non-null  int64  
 2   revenue         10729 non-null  int64  
 3   original_title  10729 non-null  object 
 4   cast            10729 non-null  object 
 5   director        10729 non-null  object 
 6   overview        10729 non-null  object 
 7   runtime         10729 non-null  int64  
 8   genres          10729 non-null  object 
 9   vote_count      10729 non-null  int64  
 10  vote_average    10729 non-null  float64
 11  release_year    10729 non-null  int64  
 12  budget_adj      10729 non-null  float64
 13  revenue_adj     10729 non-null  float64
dtypes: float64(4), int64(5), object(5)
memory usage: 1.2+ MB
```

The next step is to split the genre column. Most of the movies have multiple genres and is seperated by "|" . In most of our analysis we will need the individual genre

In [14]:
```
# to split the genres
filtered_df = filtered_df.assign(genres=df['genres'].str.split('|')).explode('genres')
filtered_df
```

Out[14]:

	popularity	budget	revenue	original_title	cast	director	overview
0	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	Twenty-two years after the events of Jurassic ...
1	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	Twenty-two years after the events of Jurassic ...
2	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	Twenty-two years after the events of Jurassic ...
3	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	Twenty-two years after the events of Jurassic ...
4	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	George Miller	An apocalyptic story set in the furthest reach...
...	...	...	...	...	...	...	...
26745	0.065141	0	0	Beregis Avtomobilya	Innokentiy Smoktunovskiy|Oleg Efremov|Georgi Z...	Eldar Ryazanov	An insurance agent who moonlights as a carthie...
26746	0.065141	0	0	Beregis Avtomobilya	Innokentiy Smoktunovskiy|Oleg Efremov|Georgi Z...	Eldar Ryazanov	An insurance agent who moonlights as a carthie...
26747	0.064317	0	0	What's Up, Tiger Lily?	Tatsuya Mihashi|Akiko Wakabayashi|Mie Hama|Joh...	Woody Allen	In comic Woody Allen's film debut, he took the...
26748	0.064317	0	0	What's Up, Tiger Lily?	Tatsuya Mihashi|Akiko Wakabayashi|Mie Hama|Joh...	Woody Allen	In comic Woody Allen's film debut, he took the...

	popularity	budget	revenue	original_title	cast	director	overview
26749	0.035919	19000	0	Manos: The Hands of Fate	Harold P. Warren|Tom Neyman|John Reynolds|Dian...	Harold P. Warren	A family gets lost on the road and stumbles up...

26750 rows × 14 columns

The breakdown of the above code

filtered_df = #reassigning the output into the same dataframe

filtered_df.assign(genres=df['genres'].str.split('|')). # this splits the string in the genres column by "|" then assignes it to genres

.explode('genres')#This tell it to add the new values as rows with the other corresponding information

.reset_index(drop=True) #This tells the dataframe to reset its index to accommodate the new entry

In [15]:
```
#To show the current status of our dataframe
filtered_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26750 entries, 0 to 26749
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   popularity      26750 non-null  float64
 1   budget          26750 non-null  int64  
 2   revenue         26750 non-null  int64  
 3   original_title  26750 non-null  object 
 4   cast            26750 non-null  object 
 5   director        26750 non-null  object 
 6   overview        26750 non-null  object 
 7   runtime         26750 non-null  int64  
 8   genres          26750 non-null  object 
 9   vote_count      26750 non-null  int64  
 10  vote_average    26750 non-null  float64
 11  release_year    26750 non-null  int64  
 12  budget_adj      26750 non-null  float64
 13  revenue_adj     26750 non-null  float64
dtypes: float64(4), int64(5), object(5)
memory usage: 2.9+ MB
```

I am done with the cleaning of the data set. i will begin my exploratory data analysis to answer the questions i stated at the introduction which are:

1. Which genres are the most popular from year to year?

2. Does the popularity of a movie have any effect on the revenue the movie generates
3. Which genre of movie produces the highest revenue?
4. Does the average vote collated for each genre have a significant effect on the revenue?

Exploratory Data Analysis

> **Tip**: Now that you've trimmed and cleaned your data, you're ready to move on to exploration. Compute statistics and create visualizations with the goal of addressing the research questions that you posed in the Introduction section. It is recommended that you be systematic with your approach. Look at one variable at a time, and then follow it up by looking at relationships between variables.

How many movies are in each genre?

In [16]:
```
#Firstly we want to find out what genres we have in out dataframe
filtered_df['genres'].unique()
```

Out[16]:
```
array(['Action', 'Adventure', 'Science Fiction', 'Thriller', 'Fantasy',
       'Crime', 'Western', 'Drama', 'Family', 'Animation', 'Comedy',
       'Mystery', 'Romance', 'War', 'History', 'Music', 'Horror',
       'Documentary', 'TV Movie', 'Foreign'], dtype=object)
```

This show us that we have 20 genres of movies present in our dataframe. We used unique() to pull out a single instance of each genre

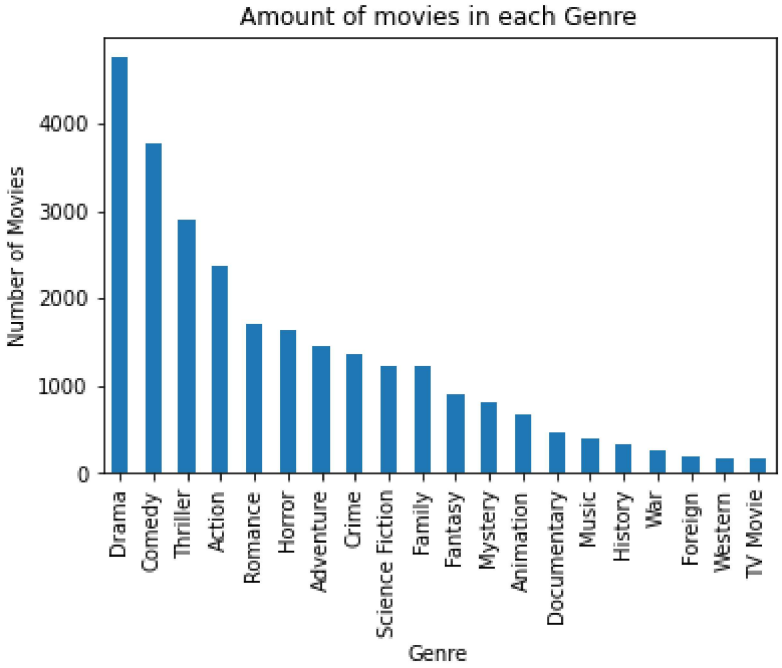
In [17]:
```
#To know how many movies are in each genre
filtered_df['genres'].value_counts()
```

Out[17]:
```
Drama              4746
Comedy             3774
Thriller           2902
Action             2376
Romance            1707
Horror             1636
Adventure          1465
Crime              1353
Science Fiction    1221
Family             1214
Fantasy             908
Mystery             808
Animation           664
Documentary         470
Music               398
History             330
War                 268
Foreign             184
Western             164
TV Movie            162
Name: genres, dtype: int64
```

This shows the number of movies that are assigned to each genre. I am going to plot a bar chart to visualize it

In [18]:
```
#This is to know show many movies are in each genre
filtered_df['genres'].value_counts().plot(kind ='bar', xlabel='Genre',ylabel='Number o
```

Out[18]:
```
<AxesSubplot:title={'center':'Amount of movies in each Genre'}, xlabel='Genre', ylabe
l='Number of Movies'>
```



The above result shows the amount of movies in each genre. Drama genre has the highest amount of movies while TV Movie genre has the lowest.

Which genres are the most popular from year to year?

In [19]:
```
# This is to show the genre available in each year according to their average populari
filtered_df.groupby(['release_year','genres']).mean('popularity')
```

Out[19]:

release_year	genres	popularity	budget	revenue	runtime	vote_count	vote_avera
1960	Action	0.590724	1.750000e+06	8.113125e+06	137.000000	65.875000	6.050(
	Adventure	0.700981	5.500000e+05	9.810000e+05	124.200000	82.000000	6.680(
	Comedy	0.396000	7.537500e+05	6.012500e+06	98.625000	44.875000	6.312!
	Crime	0.346479	0.000000e+00	0.000000e+00	119.000000	25.500000	5.900(
	Drama	0.566305	1.215919e+06	9.461538e+06	133.076923	138.153846	6.276!
...	...	...	...	...	...	...	
2015	Science Fiction	2.297221	2.918369e+07	1.144888e+08	96.809524	698.500000	5.638(
	TV Movie	0.260574	1.500000e+05	0.000000e+00	86.250000	25.700000	6.100(
	Thriller	1.401877	1.242135e+07	4.529045e+07	98.356725	343.403509	5.471:
	War	1.284511	2.377778e+07	7.845509e+07	104.000000	384.222222	6.222
	Western	3.178796	4.196667e+07	1.148233e+08	124.666667	1178.666667	6.316(

1045 rows × 8 columns

The groupby function groups the dataframe according to the year then groups it according to the genres and provides the mean of the grouping across all numerical columns

In [20]:
```
#This is to show the most popular genre from year to year
d = filtered_df.groupby(["release_year"]).apply(lambda x: x.nlargest(1, "popularity"))
d
```

Out[20]:

	popularity	budget	revenue	original_title	cast	director	
0	2.610362	806948	32000000	Psycho	Anthony Perkins|Vera Miles|John Gavin|Janet Le...	Alfred Hitchcock	I r cle
1	2.631987	4000000	215880014	One Hundred and One Dalmatians	Rod Taylor|J. Pat O'Malley|Betty Lou Gerson|Ma...	Clyde Geronimi|Hamilton Luske|Wolfgang Reitherman	Wh of pu
2	3.170651	1100000	59600000	Dr. No	Sean Connery|Ursula Andress|Joseph Wiseman|Jac...	Terence Young	I laur Jar
3	2.508235	2500000	78898765	From Russia With Love	Sean Connery|Daniela Bianchi|Lotte Lenya|Rober...	Terence Young	Ag ba in
4	3.153791	3500000	124900000	Goldfinger	Sean Connery|Honor Blackman|Gert FrÃ¶be|Shirle...	Guy Hamilton	Spe (cc
5	1.910465	11000000	141195658	Thunderball	Sean Connery|Claudine Auger|Adolfo Celi|Lucian...	Terence Young	A org has tw
6	1.227582	315000	0	How the Grinch Stole Christmas!	Boris Karloff|June Foray|Thurl Ravenscroft|Dal...	Chuck Jones|Ben Washam	E ha irri
7	2.550704	4000000	205843612	The Jungle Book	Phil Harris|Sebastian Cabot|Louis Prima|George...	Wolfgang Reitherman	T Boo the do
8	3.309196	12000000	56715371	2001: A Space Odyssey	Keir Dullea|Douglas Rain|Gary Lockwood|William...	Stanley Kubrick	I m obje
9	1.778746	7000000	81974493	On Her Majesty's Secret Service	George Lazenby|Diana Rigg|Telly Savalas|Gabrie...	Peter R. Hunt	Jar arc Ern

	popularity	budget	revenue	original_title	cast	director	
10	1.936962	4000000	55675257	The Aristocats	Phil Harris|Sterling Holloway|Scatman Crothers...	Wolfgang Reitherman	B I
11	3.072555	2200000	26589000	A Clockwork Orange	Malcolm McDowell|Patrick Magee|Adrienne Corri|...	Stanley Kubrick	Th a tou in
12	5.738034	6000000	245066411	The Godfather	Marlon Brando|Al Pacino|James Caan|Richard S. ...	Francis Ford Coppola	Spa year chr
13	2.272486	15000000	32056467	Robin Hood	Brian Bedford|Phil Harris|Peter Ustinov|Pat Bu...	Wolfgang Reitherman	R
14	3.264571	13000000	47542841	The Godfather: Part II	Al Pacino|Robert Duvall|Diane Keaton|Robert De...	Francis Ford Coppola	c sa
15	3.258151	3000000	108981275	One Flew Over the Cuckoo's Nest	Jack Nicholson|Louise Fletcher|Danny DeVito|Wi...	MiloÅ¡ Forman	ser fo
16	2.582657	1300000	28262574	Taxi Driver	Robert De Niro|Cybill Shepherd|Harvey Keitel|J...	Martin Scorsese	A Viet
17	12.037933	11000000	775398007	Star Wars	Mark Hamill|Harrison Ford|Carrie Fisher|Peter ...	George Lucas	Prin is hos
18	1.697618	6000000	181813770	Grease	John Travolta|Olivia Newton-John|Stockard Chan...	Randal Kleiser	A S
19	4.935897	11000000	104931801	Alien	Sigourney Weaver|Tom Skerritt|Veronica Cartwri...	Ridley Scott	I retu co

	popularity	budget	revenue	original_title	cast	director	
20	5.488441	18000000	538400000	The Empire Strikes Back	Mark Hamill|Harrison Ford|Carrie Fisher|Billy ...	Irvin Kershner	cor Sky
21	4.578300	18000000	389925971	Raiders of the Lost Ark	Harrison Ford|Karen Allen|Paul Freeman|Ronald ...	Steven Spielberg	Jone twe
22	4.215642	28000000	32768670	Blade Runner	Harrison Ford|Rutger Hauer|Sean Young|Edward J...	Ridley Scott	In t Los
23	4.828854	32350000	572700000	Return of the Jedi	Mark Hamill|Harrison Ford|Carrie Fisher|Billy ...	Richard Marquand	lea thei f
24	4.831966	6400000	78371200	The Terminator	Arnold Schwarzenegger|Michael Biehn|Linda Hami...	James Cameron	In ap
25	6.095293	19000000	381109762	Back to the Future	Michael J. Fox|Christopher Lloyd|Lea Thompson|...	Robert Zemeckis	Ma ac
26	2.485419	18500000	131060248	Aliens	Sigourney Weaver|Carrie Henn|Michael Biehn|Lan...	James Cameron	fo sa
27	3.474728	18000000	98235548	Predator	Arnold Schwarzenegger|Carl Weathers|Elpidia Ca...	John McTiernan	D his cor are
28	3.777441	28000000	140767956	Die Hard	Bruce Willis|Alan Rickman|Alexander Godunov|Bo...	John McTiernan	N N
29	4.143585	40000000	211343479	The Little Mermaid	Jodi Benson|Christopher Daniel Barnes|Jason Ma...	Ron Clements|John Musker	Thi a sto

	popularity	budget	revenue	original_title	cast	director	
30	2.679627	65000000	261317921	Total Recall	Arnold Schwarzenegger|Sharon Stone|Rachel Tico...	Paul Verhoeven	Cor disc
31	3.852269	25000000	377350553	Beauty and the Beast	Paige O'Hara|Robby Benson|Richard White|Jerry ...	Gary Trousdale|Kirk Wise	F ac
32	4.586426	1200000	14661007	Reservoir Dogs	Harvey Keitel|Tim Roth|Michael Madsen|Chris Pe...	Quentin Tarantino	A ir in
33	2.571339	14600000	70906973	Groundhog Day	Bill Murray|Andie MacDowell|Chris Elliott|Step...	Harold Ramis	A n wea a
34	8.093754	8000000	213928762	Pulp Fiction	John Travolta|Samuel L. Jackson|Uma Thurman|Br...	Quentin Tarantino	A phil
35	4.765359	33000000	327311859	Se7en	Brad Pitt|Morgan Freeman|Gwyneth Paltrow|John ...	David Fincher	c c
36	4.480733	75000000	816969268	Independence Day	Will Smith|Bill Pullman|Jeff Goldblum|Mary McD...	Roland Emmerich	Or g m en
37	6.668990	0	0	Eddie Izzard: Glorious	Eddie Izzard|Mac McDonald|Rhona Mitra	Peter Richardson	rou tra
38	4.180540	60000000	264118201	The Truman Show	Jim Carrey|Laura Linney|Noah Emmerich|Natascha...	Peter Weir	B t "Th
39	8.947905	63000000	100853753	Fight Club	Edward Norton|Brad Pitt|Meat Loaf|Jared Leto|H...	David Fincher	A tir i sl

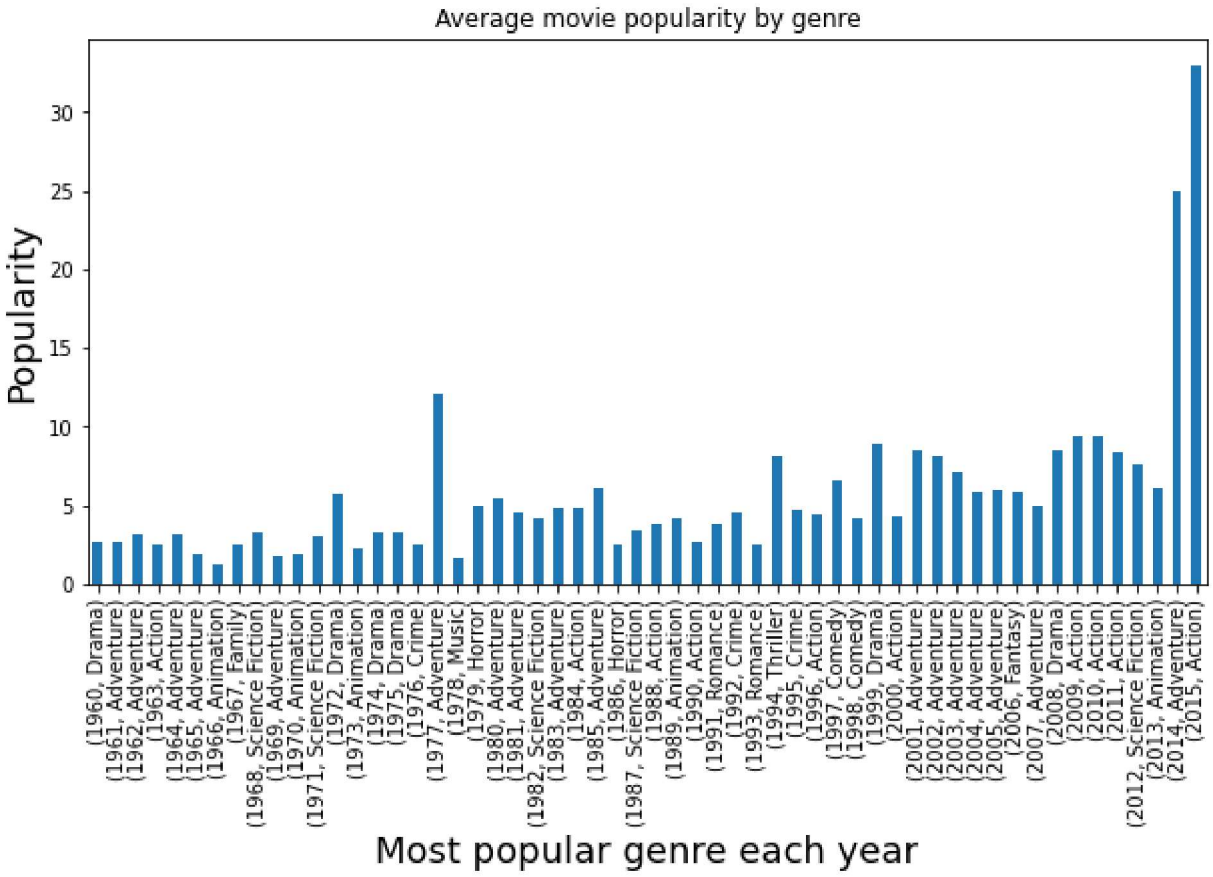
	popularity	budget	revenue	original_title	cast	director	
40	4.271452	103000000	457640427	Gladiator	Russell Crowe|Joaquin Phoenix|Connie Nielsen|O...	Ridley Scott	s ba
41	8.575419	93000000	871368364	The Lord of the Rings: The Fellowship of the Ring	Elijah Wood|Ian McKellen|Viggo Mortensen|Liv T...	Peter Jackson	hob inhe
42	8.095275	79000000	926287400	The Lord of the Rings: The Two Towers	Elijah Wood|Ian McKellen|Viggo Mortensen|Liv T...	Peter Jackson	F tr N
43	7.122455	94000000	1118888979	The Lord of the Rings: The Return of the King	Elijah Wood|Ian McKellen|Viggo Mortensen|Liv T...	Peter Jackson	A re tl the
44	5.827781	130000000	789804554	Harry Potter and the Prisoner of Azkaban	Daniel Radcliffe|Rupert Grint|Emma Watson|Gary...	Alfonso CuarÃ³n	H H I
45	5.939927	150000000	895921036	Harry Potter and the Goblet of Fire	Daniel Radcliffe|Rupert Grint|Emma Watson|Ralp...	Mike Newell	Ha H
46	5.838503	50000000	111340801	Underworld: Evolution	Kate Beckinsale|Scott Speedman|Tony Curran|Sha...	Len Wiseman	A bet vam the
47	4.965391	300000000	961000000	Pirates of the Caribbean: At World's End	Johnny Depp|Orlando Bloom|Keira Knightley|Geof...	Gore Verbinski	be be c
48	8.466668	185000000	1001921825	The Dark Knight	Christian Bale|Michael Caine|Heath Ledger|Aaro...	Christopher Nolan	sta
49	9.432768	237000000	2781505847	Avatar	Sam Worthington|Zoe Saldana|Sigourney Weaver|S...	James Cameron	In p Mar

	popularity	budget	revenue	original_title	cast	director	
50	9.363643	160000000	825500000	Inception	Leonardo DiCaprio|Joseph Gordon-Levitt|Ellen P...	Christopher Nolan	sk
51	8.411577	0	0	Underworld: Endless War	Trevor Devall|Brian Dobson|Paul Dobson|Laura H...	Juno John Lee	Un En col
52	7.637767	220000000	1519557910	The Avengers	Robert Downey Jr.|Chris Evans|Mark Ruffalo|Chr...	Joss Whedon	un eme th
53	6.112766	150000000	1274219009	Frozen	Kristen Bell|Idina Menzel|Jonathan Groff|Josh ...	Chris Buck|Jennifer Lee	
54	24.949134	165000000	621752480	Interstellar	Matthew McConaughey|Jessica Chastain|Anne Hath...	Christopher Nolan	In ac
55	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	Tw y the

I assigned the output to a new data frame so that i do not change the data in my filtered dataframe. The output also shows the most popular movie genre and the movie each year according to the genre that has the highest popularity

In [21]:
```
# This is to define a function that i will be using multiple times during my analysis.
def df_bar(counts, xlabel, ylabel, title):
    counts.plot(kind ='bar', title=title, figsize=(10,5))
    plt.ylabel(ylabel, fontsize=18)
    plt.xlabel(xlabel, fontsize=18)
    plt.show()
```

In [22]:
```
# This is to show the visualization of the result. I am calling the function i defined
counts=d.groupby(['release_year','genres']).max()['popularity']
df_bar(counts,'Most popular genre each year','Popularity','Average movie popularity by
```



The above bar chart displays the year and its most popular genre. This shows that for each year has a genre that people love to watch the most.

Does the popularity of a movie have any effect on the revenue the movie generates

In [23]:
```
# This is to analyze if the popularity of a movie relates to its revenue
rev=filtered_df['revenue']
pop=filtered_df['popularity']
corr = pop.corr(rev)
corr
```

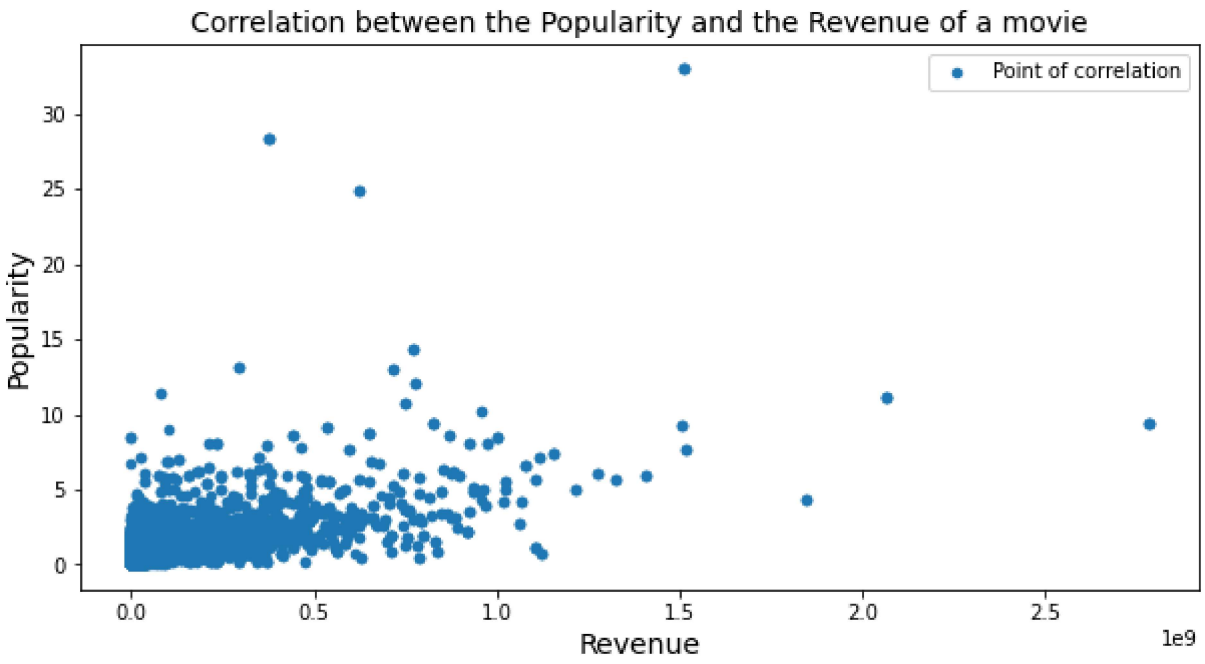
Out[23]:
```
0.6654425232361705
```

If the correlation value was equals to 1 then we can say there is a strong correlation but since the correlation value is 0.7 when abbreviated then the correlation between the popularity of a movie and the revenue the movie generates is slightly weak. This means that the popularity of a movie may not influence the revenue generated by the movie.

To visualize the correlation, i will use scatterplot

In [24]:
```
filtered_df.plot(x='revenue', y='popularity', kind='scatter',figsize=(10,5),label='Poi
plt.title('Correlation between the Popularity and the Revenue of a movie',fontsize=14)
plt.xlabel('Revenue',fontsize=14)
plt.ylabel('Popularity',fontsize=14)
plt.legend(loc='best')
```

Out[24]: <matplotlib.legend.Legend at 0x2285efa6c10>



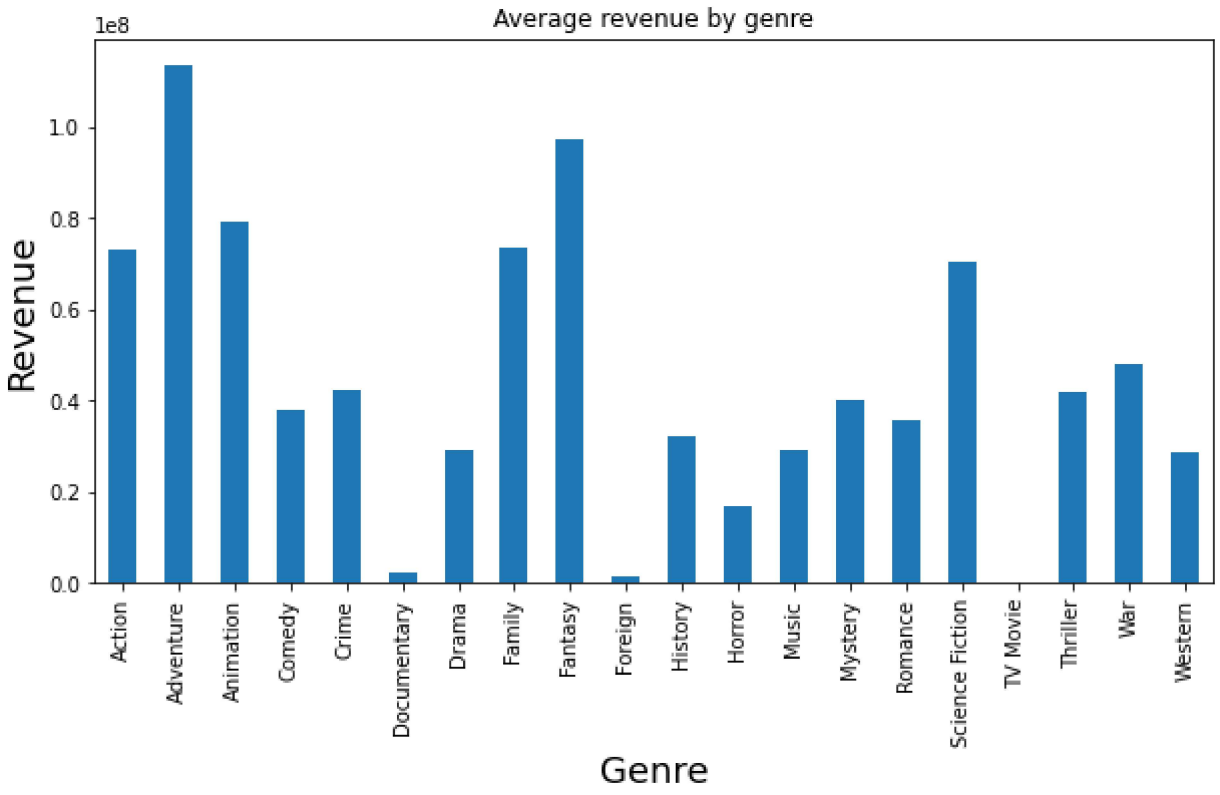
Which genre of movie produces the highest average revenue?

In [25]:
```
# This is to group the movies according to the genres and see which one has the highes
filtered_df.groupby(['genres']).mean('revenue')
```

Out[25]:

genres	popularity	budget	revenue	runtime	vote_count	vote_average	release
Action	0.929040	2.782118e+07	7.303983e+07	105.129630	394.242424	5.784891	2000.03
Adventure	1.158480	3.769746e+07	1.136012e+08	106.430034	515.130375	5.935427	1999.35
Animation	0.885913	2.437686e+07	7.922287e+07	70.664157	317.146084	6.384789	2004.06
Comedy	0.594878	1.336239e+07	3.771225e+07	96.935612	177.228670	5.901245	2000.78
Crime	0.745331	1.767686e+07	4.239998e+07	106.926829	279.002956	6.124982	1999.48
Documentary	0.188172	5.958754e+05	2.182688e+06	103.970213	36.200000	6.898511	2008.07
Drama	0.592844	1.191304e+07	2.931604e+07	110.516224	183.027181	6.163611	2000.93
Family	0.794195	2.367775e+07	7.344748e+07	90.175453	275.586491	5.986903	2000.76
Fantasy	1.000166	3.289441e+07	9.716224e+07	101.169604	424.170705	5.856057	2000.23
Foreign	0.191552	1.442607e+06	1.493731e+06	107.016304	16.722826	5.978804	2001.23
History	0.582103	1.879910e+07	3.239981e+07	134.336364	185.466667	6.411818	1997.43
Horror	0.465370	6.230335e+06	1.683309e+07	94.426039	120.127139	5.337714	2001.11
Music	0.495907	9.656181e+06	2.928814e+07	105.964824	127.052764	6.468844	2000.14
Mystery	0.691137	1.615917e+07	4.031712e+07	106.069307	237.538366	5.946535	1999.83
Romance	0.593384	1.256553e+07	3.579007e+07	107.053310	166.504979	6.042707	2000.40
Science Fiction	1.007173	2.513630e+07	7.060012e+07	99.811630	439.832924	5.657576	1999.90
TV Movie	0.272252	2.759259e+05	2.592593e+05	93.271605	34.716049	5.741975	2004.41
Thriller	0.742534	1.723631e+07	4.180031e+07	103.288077	255.894555	5.750310	2001.68
War	0.732346	2.104780e+07	4.796045e+07	127.753731	272.514925	6.296642	1995.98
Western	0.594216	1.908980e+07	2.874291e+07	114.871951	206.890244	6.083537	1986.92

In [26]:
```
# To understand the output above we will do a visualization in bar chart
counts=filtered_df.groupby(['genres']).mean()['revenue']
df_bar(counts,'Genre','Revenue','Average revenue by genre')
```



From the bar chart above i can deduce that the movie genre "Adventure" has the highest average revenue across all the genres

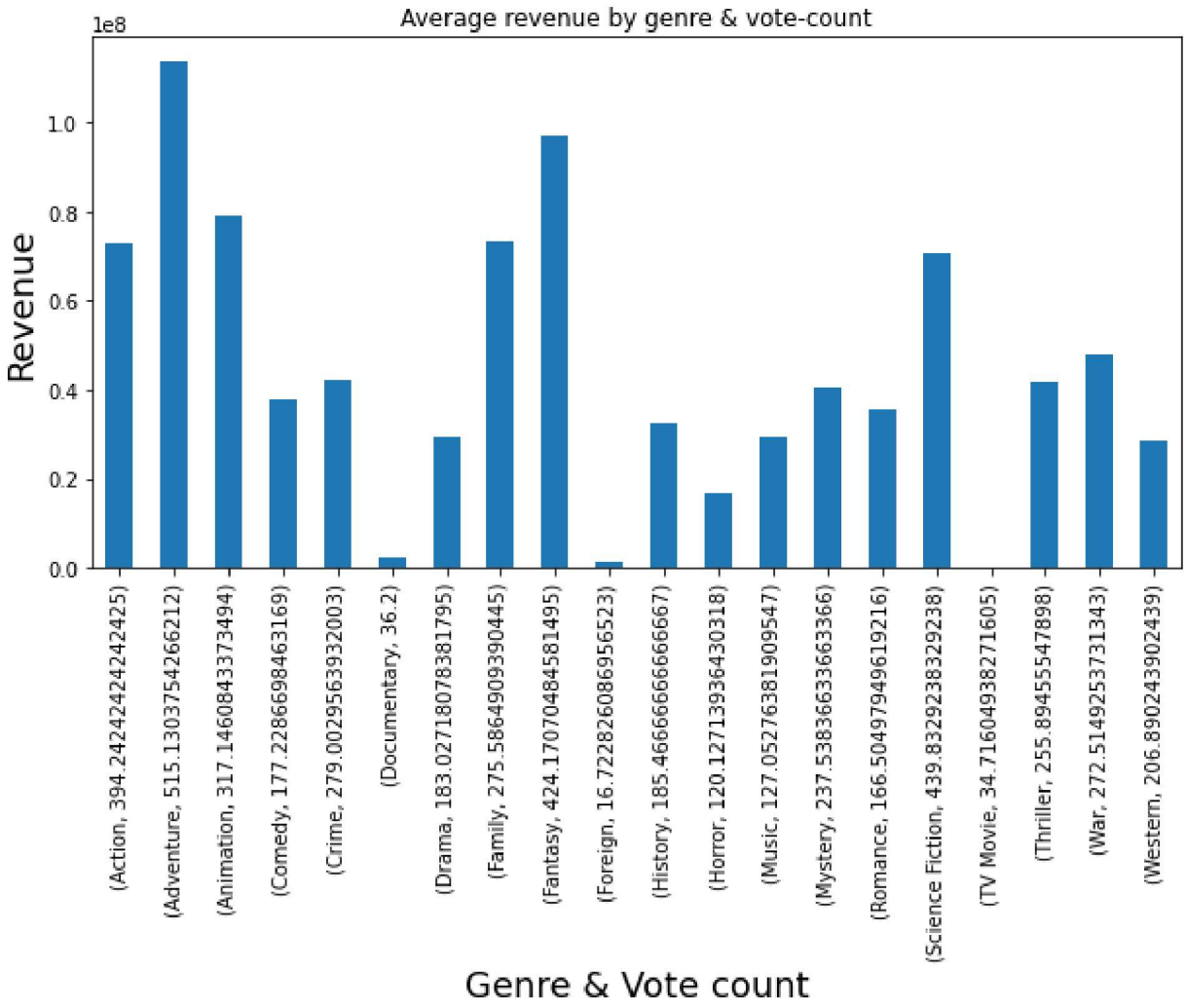
Does the average vote collated for each genre have a significant effect on the revenue?

In [27]:
```
#This shows the average votes collated for each year
d_vote = filtered_df.groupby(["genres"]).mean(['vote_count'])
d_vote
```

Out[27]:

genres	popularity	budget	revenue	runtime	vote_count	vote_average	release
Action	0.929040	2.782118e+07	7.303983e+07	105.129630	394.242424	5.784891	2000.03
Adventure	1.158480	3.769746e+07	1.136012e+08	106.430034	515.130375	5.935427	1999.35
Animation	0.885913	2.437686e+07	7.922287e+07	70.664157	317.146084	6.384789	2004.06
Comedy	0.594878	1.336239e+07	3.771225e+07	96.935612	177.228670	5.901245	2000.78
Crime	0.745331	1.767686e+07	4.239998e+07	106.926829	279.002956	6.124982	1999.48
Documentary	0.188172	5.958754e+05	2.182688e+06	103.970213	36.200000	6.898511	2008.07
Drama	0.592844	1.191304e+07	2.931604e+07	110.516224	183.027181	6.163611	2000.93
Family	0.794195	2.367775e+07	7.344748e+07	90.175453	275.586491	5.986903	2000.76
Fantasy	1.000166	3.289441e+07	9.716224e+07	101.169604	424.170705	5.856057	2000.23
Foreign	0.191552	1.442607e+06	1.493731e+06	107.016304	16.722826	5.978804	2001.23
History	0.582103	1.879910e+07	3.239981e+07	134.336364	185.466667	6.411818	1997.43
Horror	0.465370	6.230335e+06	1.683309e+07	94.426039	120.127139	5.337714	2001.11
Music	0.495907	9.656181e+06	2.928814e+07	105.964824	127.052764	6.468844	2000.14
Mystery	0.691137	1.615917e+07	4.031712e+07	106.069307	237.538366	5.946535	1999.83
Romance	0.593384	1.256553e+07	3.579007e+07	107.053310	166.504979	6.042707	2000.40
Science Fiction	1.007173	2.513630e+07	7.060012e+07	99.811630	439.832924	5.657576	1999.90
TV Movie	0.272252	2.759259e+05	2.592593e+05	93.271605	34.716049	5.741975	2004.41
Thriller	0.742534	1.723631e+07	4.180031e+07	103.288077	255.894555	5.750310	2001.68
War	0.732346	2.104780e+07	4.796045e+07	127.753731	272.514925	6.296642	1995.98
Western	0.594216	1.908980e+07	2.874291e+07	114.871951	206.890244	6.083537	1986.92

In [28]:
```
# To show if the average vote collated for each genre have a significant effect on the
counts=d_vote.groupby(['genres','vote_count']).mean()['revenue']
df_bar(counts,'Genre & Vote count','Revenue','Average revenue by genre & vote-count')
```



This shows that the vote count for each genre has an effect on the revenue of each genre.Those genres with low votes have low revenues while those with high votes have high revenue

Conclusions

According to all the analysis that i have done on the Movie data set, the analysis gives an insight to

1. The most popular genre of movie each year
2. The possible effect that the popularity of a movie can have on its revenue
3. The movie genre that has the highest revenue which is Adventure
4. The number of votes that are accruded to a movie genre having an effect on the revenue generated

Some of the limitation of this data set is that the null values were many and i couldn't remove them from the data set as they were vital tomy analysis so i kept them in a seperate dataframe.

Secondly as i had a long range of years to work with, i summarized some of my findings as i could not list them out one by one.

The websites that i referred to in the course of my analysis are

https://stackoverflow.com https://www.sfu.ca https://datatofish.com › Python pandas.pydata.org www.tutorialspoint.com classroom.udacity.com

In [ ]: