Project: Investigate a Dataset

Table of Contents

Introduction

This data set contains information about 10,000 movies collected from The Movie Database (TMDb), including user ratings and revenue,budget,revenue,original_title,cast etc.This project is associated with using this dataset as input and draw meaningful observations.

- **Importing the necessary libraries such as numpy and pandas for calculations and manipulations of the csv.For plotting matplotlib and seaborn has been imported**

In [1]:
```
# Import Statements
import pandas as pd
import numpy as np
import csv
import seaborn as sns
from datetime import datetime
import matplotlib.pyplot as plt
% matplotlib inline
```

Data Wrangling

In this section of the report, we will load in the data, check for cleanliness, and then trim and clean the dataset for analysis.

- **Reading the csv and viewing few rows to have a look at the dataframe**

In [2]:
```
df = pd.read_csv('tmdb-movies.csv')  ##Reading the csv file
df.head(3)   ##Viewing the first few columns
```

Out[2]:

	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.

3 rows × 21 columns

Getting the Size of the data

In [428]:
```
df.shape
```

Out[428]:
```
(10866, 21)
```

Finding info about the data to get an estimate about the number of null values in diffrerent columns

In [429]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
id                      10866 non-null int64
imdb_id                 10856 non-null object
popularity              10866 non-null float64
budget                  10866 non-null int64
revenue                 10866 non-null int64
original_title          10866 non-null object
cast                    10790 non-null object
homepage                2936 non-null object
director                10822 non-null object
tagline                 8042 non-null object
keywords                9373 non-null object
overview                10862 non-null object
runtime                 10866 non-null int64
genres                  10843 non-null object
production_companies    9836 non-null object
release_date            10866 non-null object
vote_count              10866 non-null int64
vote_average            10866 non-null float64
release_year            10866 non-null int64
budget_adj              10866 non-null float64
revenue_adj             10866 non-null float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

From the above result we can see there are certain columns which have null values such as imdb_id,cast,homepage,director,overview,production companies.So we need to drop those rows.

Getting a brief description about the dataset

In [430]:
```
df.describe()
```

Out[430]:

	id	popularity	budget	revenue	runtime	vote_count	vo
count	10866.000000	10866.000000	1.086600e+04	1.086600e+04	10866.000000	10866.000000	1(
mean	66064.177434	0.646441	1.462570e+07	3.982332e+07	102.070863	217.389748	
std	92130.136561	1.000185	3.091321e+07	1.170035e+08	31.381405	575.619058	
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.000000	
25%	10596.250000	0.207583	0.000000e+00	0.000000e+00	90.000000	17.000000	
50%	20669.000000	0.383856	0.000000e+00	0.000000e+00	99.000000	38.000000	
75%	75610.000000	0.713817	1.500000e+07	2.400000e+07	111.000000	145.750000	
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	

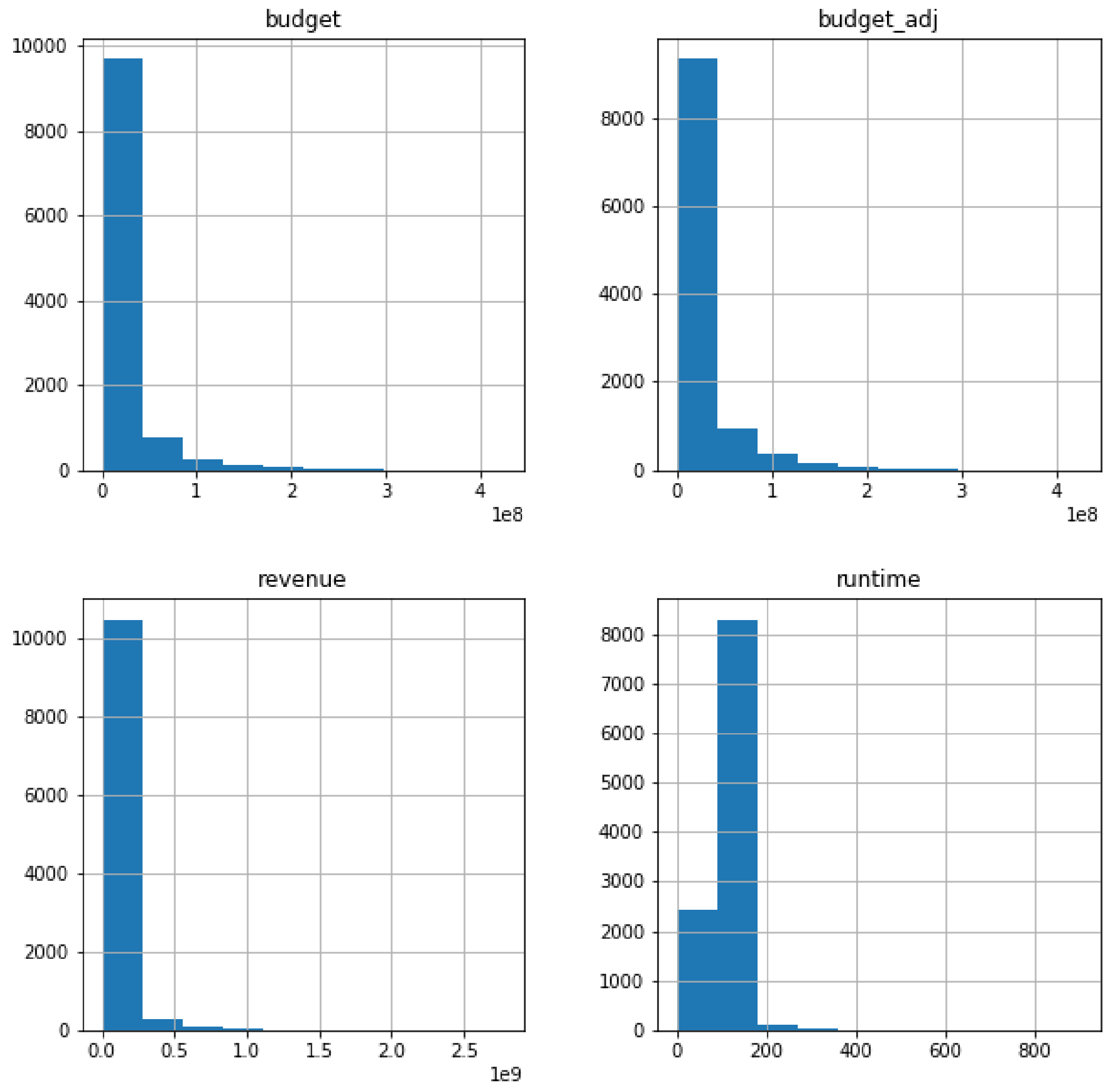
Finding datatypes of different columns to check whether any coulnm has wrong datatype

In [431]:
```
df.dtypes
```

Out[431]:
```
id                        int64
imdb_id                  object
popularity              float64
budget                    int64
revenue                   int64
original_title           object
cast                     object
homepage                 object
director                 object
tagline                  object
keywords                 object
overview                 object
runtime                   int64
genres                   object
production_companies     object
release_date             object
vote_count                int64
vote_average            float64
release_year              int64
budget_adj              float64
revenue_adj             float64
dtype: object
```

As we can see above release date of of object type,which I guess is wrong.This should be of datetime type.So we need to convert it to the correct type

```
In [9]: #Histogram for budget,runtime,revenue,budget_adj
        a=df[['budget','runtime','revenue','budget_adj']].hist(figsize=(10,10));
```



- **The histograms above shows the different parameters such as budget,revenue,runtime,budget_adj for the movies listed in the dataframe.We can see that budget required to make a movie is around 0.5e8.Also we can see that runtime for average movies is 90-150 min**

Data Cleaning

Update the datatypes

- **Converting the data types into suitable types.Since date cannot be str type.so we convert it top datetime type.**

In [378]:
```
#converting the data types into suitable types
df['release_date'] = pd.to_datetime(df['release_date'])
```

Droping columns

- **We drop all the columns which we do not need for any manupulations i.e from which data we cannot make out any thing.**

In [379]:
```
#creating a list of columb to be deleted
delete_col=[ 'imdb_id','revenue_adj', 'homepage', 'keywords', 'overview', 'pro
duction_companies','tagline']

#deleting the un-necessary columns
df= df.drop(delete_col,1)

#Viewing the new dataset
df.head(2)
```

Out[379]:

	id	popularity	budget	revenue	original_title	cast	director	runtim
0	135397	32.985763	150000000.0	1.513529e+09	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	Colin Trevorrow	12
1	76341	28.419936	150000000.0	3.784364e+08	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	George Miller	12

We check if our data contains some duplicates & null values.If such values are found we remove them because if we try working on incomplete data we might end up getting reverse result than expected.

In [358]:
```
#checking for the duplicate rows
sum(df.duplicated())
```

Out[358]: 1

The out 1 indicates that there one duplicate row in the dataframe which we need to remove.

In [359]:
```
#Removing the duplicate rows
df.drop_duplicates(keep= 'last',inplace = True)
```

Looking for the number of rows having null values

In [360]:
```
df.isnull().sum().sort_values(ascending = False)
```

Out[360]:
```
cast              76
director          44
genres            23
budget_adj         0
release_year       0
vote_average       0
vote_count         0
release_date       0
runtime            0
original_title     0
revenue            0
budget             0
popularity         0
id                 0
dtype: int64
```

After dropping the null and dupicate values we check the shape of the dataframe to know how data loss we have had.

In [361]:
```
#Droping values having null values
df.dropna(axis = 0, how ='any',inplace = True)
df.shape
```

Out[361]:
```
(10731, 14)
```

Removing 0's from budget and the revenue columns

- **We remove rows with 0 in the the revenue or budget coulmn as either the budget of the revenue cannot be 0.If it has been mentione as 0 that means there is some ambiguity with that data.So it's better to remove that.**

- **Creating a seperate list of revenue and budget column**
- **This will replace all the value from '0' to NAN in the list**
- **Removing all the row which has NaN value in temp_list**

In [362]:
```
remove_list=['budget', 'revenue']
df[temp_list] = df[remove_list].replace(0, np.NAN)
df.dropna(subset = remove_list, inplace = True)
```

In [363]:
```
df['genres'].str[:].head(10)
```

Out[363]:
```
0    Action|Adventure|Science Fiction|Thriller
1    Action|Adventure|Science Fiction|Thriller
2           Adventure|Science Fiction|Thriller
3     Action|Adventure|Science Fiction|Fantasy
4                        Action|Crime|Thriller
5             Western|Drama|Adventure|Thriller
6    Science Fiction|Action|Thriller|Adventure
7              Drama|Adventure|Science Fiction
8            Family|Animation|Adventure|Comedy
9                    Comedy|Animation|Family
Name: genres, dtype: object
```

Diffrent genres mentioned for the movies can be Action,adventure,Science Fiction,Thriller,Drama etc.

Exploratory Data Analysis

Which are the Highest grossing movie each year ?

- **In order to find the highest grossing movie,We first sort the dataframe based on revenue.**
- **Then we groupby 'release year' so that we get the highest revenue for each movie for that particular year.**

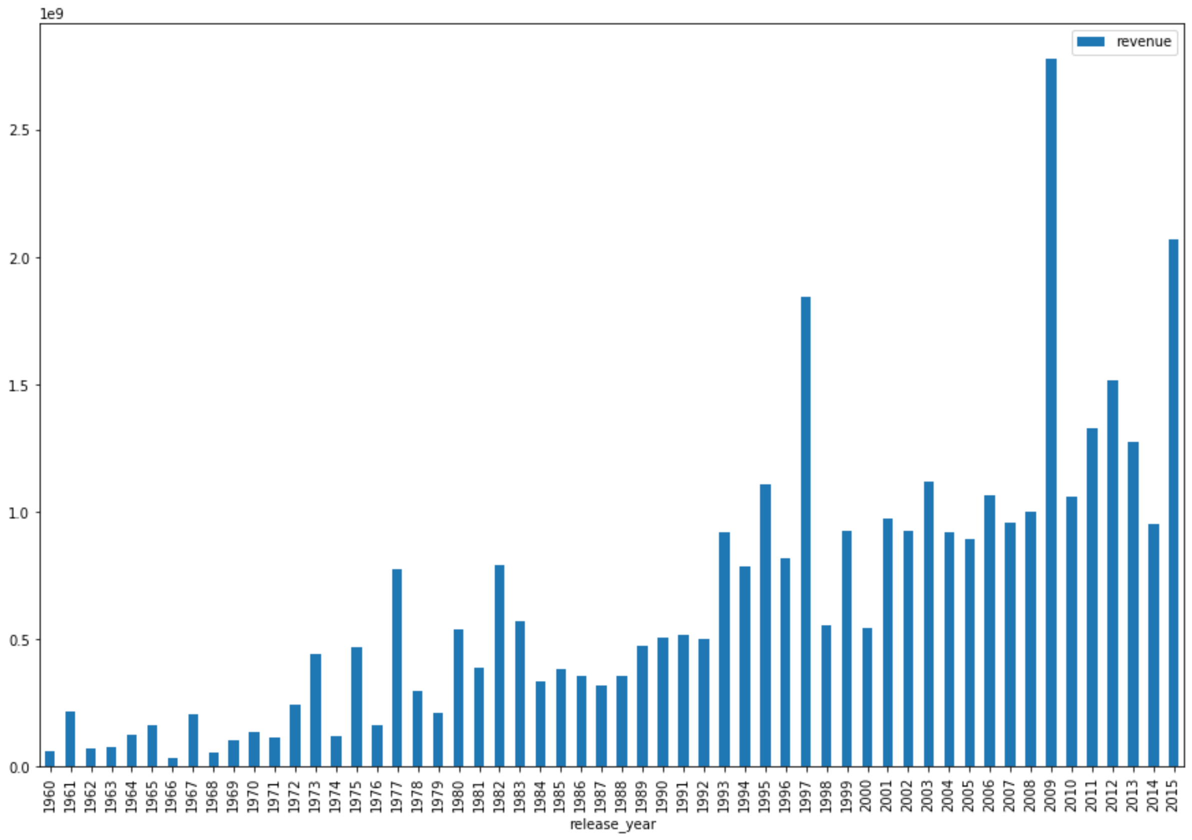
In [380]:
```
##sorting the dataframe based on revenue and then grouping by release years.

sorted_df=df.sort_values(['revenue'],ascending=False)
grouped_by_year=sorted_df.groupby('release_year').head(1)[['release_year','rev
enue','original_title']].sort_values(['release_year']).set_index('release_yea
r')
grouped_by_year.head()
```

Out[380]:

	revenue	original_title
release_year		
1960	60000000.0	Spartacus
1961	215880014.0	One Hundred and One Dalmatians
1962	70000000.0	Lawrence of Arabia
1963	78898765.0	From Russia With Love
1964	124900000.0	Goldfinger

Then we plot the grouped_by_year dataframe which contains the 'release_year' and the highest revenue for that year

In [333]:
```
grouped_by_year.plot.bar(figsize=(15,10),legend='revenue',);
```



- **From the plot above we can see the revenue generated by movies has been increasing in successiuve years.In the year 2009 max revenue was generated**

2 What is the average Movie duration, longest movie and shortest movie?

To find the average movie duration we find the mean of the 'runtime' column from the TMDb dataframe

In [381]:
```
##method to calculate the average time
def avg_runtime(column):
    run_time= df[column].mean()
    return run_time
avg_runtime('runtime')
```

Out[381]:
```
109.21745908028059
```

- **To find maximum run time we apply max() method on the 'runtime' column**
- **Then we fetch that particular row using the loc[] method**
- **Finally we display the required columns.**

In [404]:
```
#Max run time and min run time
Longest_duration=df['runtime'].max()
df.loc[(df['runtime'] == Longest_duration),['runtime','original_title']]
```

Out[404]:

	runtime	original_title
2107	338	Carlos

- **To find minimum run time we apply min() method on the 'runtime' column**
- **Then we fetch that particular row using the loc[] method**
- **Finally we display the required columns.**

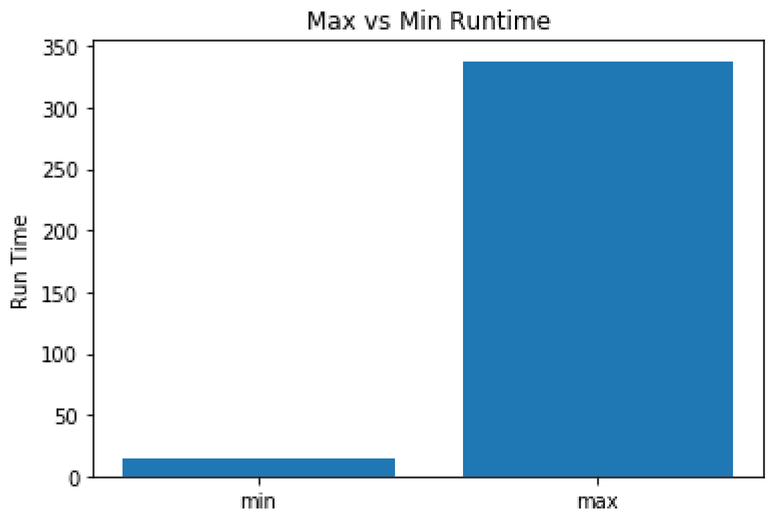
In [402]:
```
#Min run time and min run time
shortest_duration=df['runtime'].min()
df.loc[(df['runtime'] == shortest_duration),['runtime','original_title']]
```

Out[402]:

	runtime	original_title
5162	15	Kid's Story

- **First we find the maximum run time and the minimum run time.**
- **Then we use the bar() method to make the bar plot.**
- **Finally we provide the proper label for the x-axis,y-axis and the label.**

In [405]:
```
max=df['runtime'].max()

min=df['runtime'].min()
objects = ('min', 'max')
y_pos = np.arange(len(objects))
performance = [min,max]
plt.bar(y_pos, performance, align='center')
plt.xticks(y_pos, objects)
plt.ylabel('Run Time')
plt.title('Max vs Min Runtime')
plt.show()
```

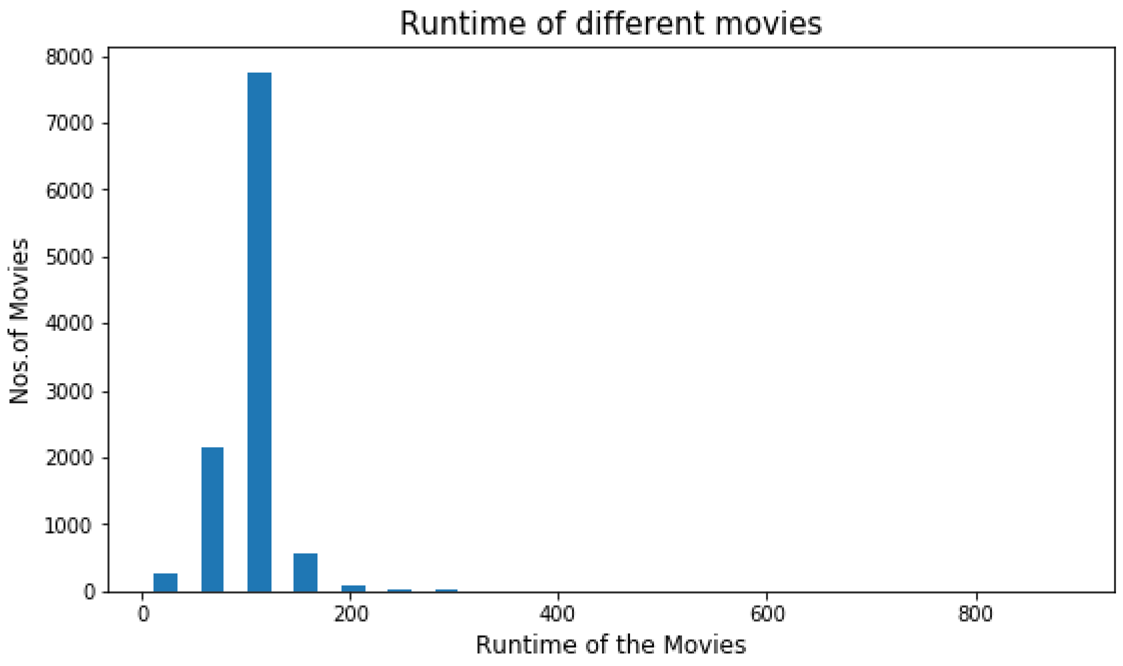


- **The plot shows the max and the min run time for the movies given in the csv**

3 Run time of different movies ?

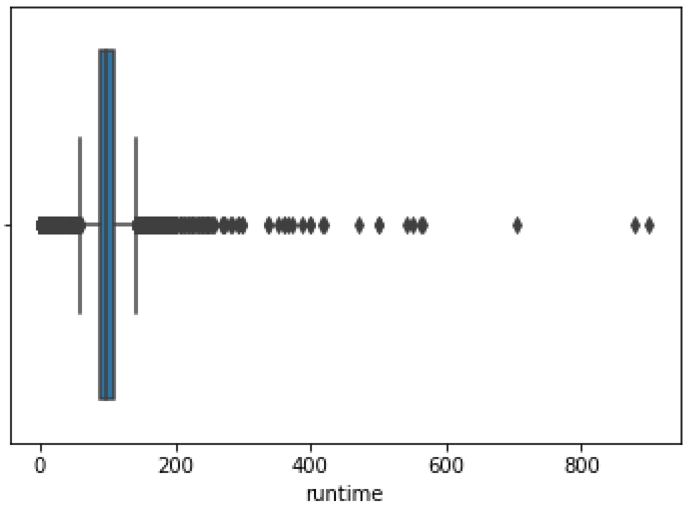
- **Here we make histogram for the rumtime to analyse the runtime for different movies.**

In [6]:
```
plt.figure(figsize=(9,5))

#On x-axis
plt.xlabel('Runtime of the Movies', fontsize = 12)
#On y-axis
plt.ylabel('Nos.of Movies', fontsize=12)
#Name of the graph
plt.title('Runtime of different movies', fontsize=15)

#giving a histogram plot
plt.hist(df['runtime'], rwidth = 0.5,bins =20)
#displays the plot
plt.show()
```



The histogram above shows that generally movies runtime is 90-150 min

In [6]:
```
##Box plot for runtime
sns.boxplot(df["runtime"]);
```



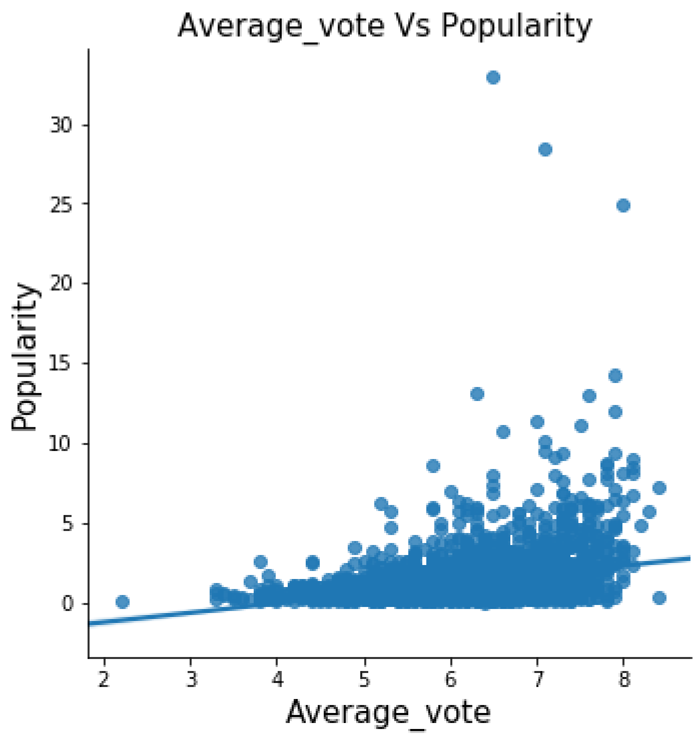
- **A box plot is added additionally to have better visualisation.**

4 Does popularity of a movie depend on avg vote ?

- **To find this dependency we try to plot the 'popularity' against the average vote.**
- **we see in the plot that with rise in average vote there is rise in the popularity on an average.**

In [383]:
```
plt.figure(figsize = (25,15)); #set a figure size

sns.lmplot(x = 'vote_average', y = 'popularity', data = df); #plot a lineplot
plt.xlabel('Average_vote', fontsize = 15);
plt.ylabel('Popularity', fontsize = 15);
plt.title('Average_vote Vs Popularity ',fontsize = 15);
```

```
<Figure size 1800x1080 with 0 Axes>
```



- **The plot above shows that movies which has higher average has greater popularity except for some outliers.**

5 Most popular and least popular movie ?

Most popular movie

- **To find the most popular movie first we find the maximum value from the popularity column.The we fetch the row corresponding to that value.Then finally we get the name of the movie from that row.**

```
In [421]: ##most popular movie
          most_popular=df['popularity'].max()
          df.loc[(df['popularity'] == most_popular),['popularity','original_title']]
```

Out[421]:

	popularity	original_title
0	32.985763	Jurassic World

From the result above we can see that Jurassic World has highest rating.

Least popular movie

- **To find the least popular movie first we find the minimum value from the popularity column.The we fetch the row corresponding to that value.Then finally we get the name of the movie from that row.**

```
In [425]: ##Least popular movie
          least_popular=df['popularity'].min()
          df.loc[(df['popularity'] == least_popular),['popularity','original_title']]
```

Out[425]:

	popularity	original_title
7268	0.001117	Born into Brothels

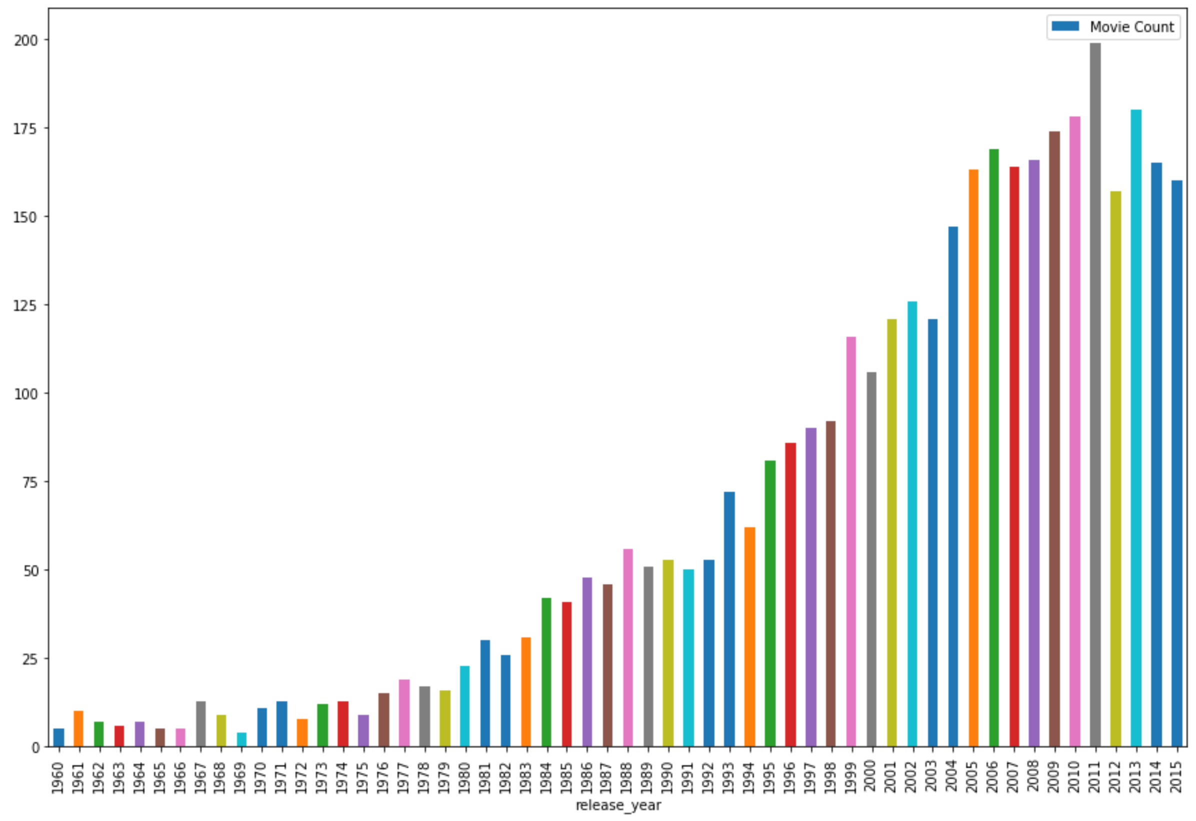
From the result above we can see that Born into Brothels has least rating.

6 Number of movies release each year ?

- **We have to first groupby the dataframe by 'release_year'**
- **Then we count the number of Ids under the 'id' column to count the number of movies.**
- **Finally we plot the values using bar plot.**

```
In [245]: #grouping by release_year and then counting the number of id under each year f
          or number of movies released.
          grouped_to_count=df.groupby('release_year').count()['id']
```

In [246]:
```
grouped_to_count.plot.bar(figsize=(15,10),legend='Movie Count',label='Movie Co
unt');
```



- **From the plot above we can conclude that Number of movies released has increased over the years,with maximum number of movies released in the year 2011**

5 Budget vs Profit ?

- **First we fetch the 'revenue' and the 'budget' from the dataframe.**
- **Then we obtain the profit by deducting the 'budget' from the 'revenue'.**
- **The we fetch the 'budget' column.**
- **Finally we plot the values.**

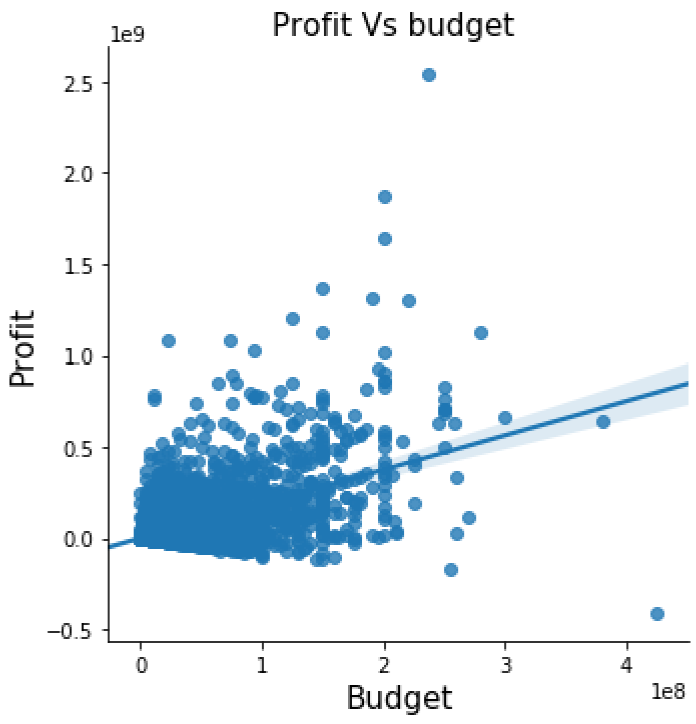
In [315]:
```
df['profit']=df['revenue']-df['budget']  ##profit=revenue-budget
temp=df[['budget','revenue','profit']]
temp.head()
```

Out[315]:

	budget	revenue	profit
0	150000000.0	1.513529e+09	1.363529e+09
1	150000000.0	3.784364e+08	2.284364e+08
2	110000000.0	2.952382e+08	1.852382e+08
3	200000000.0	2.068178e+09	1.868178e+09
4	190000000.0	1.506249e+09	1.316249e+09

In [327]:
```
plt.figure(figsize = (25,15)); #set a figure size

sns.lmplot(x = 'budget', y = 'profit', data = temp); #plot a lineplot
plt.xlabel('Budget', fontsize = 15);
plt.ylabel('Profit', fontsize = 15);
plt.title('Profit Vs budget ',fontsize = 15);
```

```
<Figure size 1800x1080 with 0 Axes>
```



- **From the plot we can see that profit increases with budget as the plot has a rising curve.Any way we can see some outliers but still we can conclude this**

Conclusions

- **Overall analysis**
  - **Initially the dataset contained 10866 rows in total but there were several rows which contained null values,after removing those rows I was left with 10731.The I removed all the unnecessary coulumns.After wrangling and cleaning the data I did my over all analysis and found that:-**
  - **Number of movies released has increased over the years,with maximum number of movies released in the year 2011**
  - **Frequent Runtimes are from 90 sec to 150 min**
  - **Average movie duration is 109.2 min**
  - **Most often it is seen that Profit increases with budget**
  - **Movies having higher average rating have greater popularity on an average.**
  - **By revenue generated**
    - Highest grossing movie is Avtar released in the year 2009.
    - Lowest grossing movie is Who's Afraid of Virginia Woolf released in the year 1966.
  - **Number of movies release each year**
    - Maximum number of movies released in the year 2014 ie 700
    - Minimum number of movies released in the year 1961 and 1969 ie 31.
  - **Based on runtime**
    - Shortest Movie =Kid's Story
    - Longest Movie =Carlos
  - **Based on popularity**
    - Most popular =Jurassic world
    - Least popular =Born into brothels

```
In [ ]:
```