# **Introduction**

Movies!! When we think movies the first words that come to our minds are action, romance, thrill and what you may. However here we want to take a cold hard statistical look at the magic called movies. We have sliced & diced the numbers, crunched few numbers. It is for you to judge if statistics comes out as ever powerful super-hero who can do no wrong or a flawed hero, hero none the less.

We have used popular and reliable website [www.imdb.com](http://www.imdb.com) (Internet Movie Database) to source our data. Data is sourced using their APIs and we were able to fetch data on movies since 1910. Movies are split across genres, language, ratings , the duration of the film and many other factors including the obvious ones like actors, director, year, revenue etc.

A commercial successful movie not only entertains audience, but also enables film companies to gain tremendous profit. A lot of factors such as good directors, story, cast along with a good budget are considerable for creating good movies.

Movie ratings and reviews at sites such as IMDb are commonly used by moviegoers to decide which movie to watch or buy next.

The objective of this exercise is to take a cold hard statistical look at the business of movie making. It will also challenge or ascertain popular beliefs on popularity, salability

# **Objective**

This project aims to analyze the movies data on the IMDb website over 100 years (released in 1910-2018) with graphics and gives an interpretation of these data.

# **Preparing the dataset**

Data is extracted using API from IMDb website, URL: <https://www.imdb.com>

**API details**

Create account and get API key

The Movie Database API has two GET endpoints

* Search ID
* Title Search

(This endpoint allows you to search movie and TV shows by title. The API returns a list of movie or TV show results based on your search query.)

The optional parameters include:

* page – return results by page (1-100)
* y – specify the year of release
* r – specify the result format (JSON or XML)
* type – specify the type of result to return. (The three options include movie, series or episode)

**For each JSON item, the API returns:**

* Title
* Release Year
* IMDb ID
* Type (movie, series, or episode)
* Movie Poster image

**API returns following (column)values**

Title ,Release Year ,Rating, Release Date, Runtime, Genre(s), Director(s), Writer(s), Actor(s), Plot Summary, Language(s), Country/Countries, Awards Won, Movie posters (URL of film image), Ratings received, Metascore, IMDb Rating IMDb Votes IMDb ID Type (movie, series, or episode) DVD info Box Office results Production company Website(s)

We are taking list of top 5000 movies since 1910<Ankit to add - how top 5000 movies selected >

**Here are the final data we are interested in:**

* Title – Movie Title
* Year – Movie released year
* Rated – IMDb rating category
* Runtime- Duration of Movie
* Genre – movie genre
* Director – director of movie
* Writer – movie writers
* Language – spoken languages in movie
* Country – country of production
* Awards – awards received
* Ratings (score out of 10)
* Revenue – movie revenue
* Budget- movie budget

**Data Cleaning**

This dataset can be fixed as follows:

1. Delete the line with the missing values
2. Fill empty fields with specific values
3. Fill empty fields with calculations

We went with option 1 & 2

# **Data analysis**

# **Visualization of data and interpretation of data**

# This study through a large volume of data, determine the following points for movies released between 2010 and 2018:

* Genre drama is the most frequent and revenue earning genre.
* Average revenue of movies increased exponentially with each decade.
* Number of movies produced and profit increased with each passing decade.
* Most movies last between 60 minutes and 120 minutes
* Movies that are well rated by public and critics make the most money
* The more the public appreciates a film, the more they vote and give a good rating
* Movies
* Movies that exceed 3 hours bring in the least money
* Add <interpretation for your charts>
* Add <interpretation for your charts>
* add
* add
* add
* add