Milestone 1

Dataset Details

Dataset 1:

Movie details using OMDB API - Getting movie details, actors, awards, DVD release date, Rating Scores from Rotten Tomatoes and IMDb, box office revenue etc. API returns 25 attributes about a movie. For 250 top movies from IMDB I am calling this API in loop to get details for every movie by passing movie name and the year of release. There are total 250 records.

Dataset 2:

Netflix movies and shows dataset from Kaggle - Netflix is one of the most popular media and video streaming platforms. They have over 8000 movies or tv shows available on their platform, as of mid-2021, they have over 200M Subscribers globally. This tabular dataset consists of listings of all the movies and tv shows available on Netflix, along with details such as - cast, directors, ratings, release year, duration, etc. This is CSV file downloaded from Kaggle.

Dataset 3:

IMDB's top 250 movie data using web scraping - IMDB ranks movies and given them score. This dataset is published on IMDB website that includes top 250 movies of all time according to them. using Python's Beatifulsoup library to webscrape and json parsing to get the information parsed out to build the dataset. There are 250 movie records.

Dataset 4:

Movie Torrents for top 100 monthly movies from Movies, TV, Music, Search, and, Download API - Movie, TV, music search and download provides API connections to download torrent links for monthly top 100 movies, monthly top 100 tv shows, monthly top 100 music videos, monthly top 100 games torrent etc.

Relationship between datasets

Dataset-1 (OMDB) and dataset-3 (IMDB) we have "imdb id" to join on.

Dataset-2 (Netflix) and Dataset-4 (Torrent) can be connected on movie title. Need to be careful here as Torrent dataset has more than one record per movie title. I am thinking to get that to one record per movie with an array of torrent links and count of torrents available as derived fields. Doing that grouping by movie titles will bring the Torremt dataset to same granularity level as that of Netflix dataset.

Above two resulting datasets can then be connected using "movie title" and "year". While doing join I will be careful about case of movie titles, any special characters or white spaces, format of year etc to make sure all our join conditions pass and get a corresponding row.

Project Subject Area

Project is about analysing details of top movies in the US. Analysis that I am interested in are -

1. Distribution of Domestic vs Foreign movies with respect to awards through time
2. Most popular genre of movies both domestic and foreign
3. Does box office revenue correlates with awards?
4. Popularity trend of shows vs movies on Netflix
5. Duration trend of movies through time - have movies been shorter through the time?
6. Popular movies by numbner of torrents created
7. Actors/Actresses by award counts through time

What will it take to accomplish 5 final project milestones?

1. Data quality improvement - i.e. outlier detection and remediation, data formatting changes etc.
2. Data integration from different sources
3. Handling data granularity while joining different data sets together - This may need grouping certain datasets while creating aggregated metrics before joining it to other datasets.
4. Establishing connection to the chosen database / RDBMS.
5. Creating table(s) to store the data gathered.
6. Load the data into tables.
7. Create all thought about data visualizations and document the trends, observations etc.
8. Capture comments wherever applicable.
9. Have saved copies of all datasets at different stages including python scripts with glimpse into data, data base tables, visualizations in python scripts or any visualization tools.
10. Push all the changes to GIT repo, provide needed links etc during final submission.

Ethical implications and challenges

Well, there is a lot of ethical implications that movies have on human mind, his/her development of thoughts, influencing changes in behavior etc. For e.g. some movies show violence in raw form which can leave one distressed and thinking about it for long time. In terms of project at hand any analysis is not an attemp to influence or provide guidance to watch certain types of movies but instead look at the trend and build understanding of this space with data.

As of now I do not see a lot of challenge with this project. Said that data needs to cleaned, integrated, and stored.

Dataset 1: Getting movie details, actors, awards, DVD release date, Rating Scores from Rotten Tomatoes and IMDb, box office revenue etc using OMDB API

Dataset Description:

The OMDb API is a RESTful web service to obtain movie information.

Data Source:

1. OMDB API accessed using python requests library - 'http://www.omdbapi.com/?apikey='+API_KEY+'&t='+title+'&y='+year

Metadata:

1. Title - Title or Name of the movie (String)
2. Year - Movie release Year (Number)
3. Rated - PG rating (String)
4. Released - Release date (Date)
5. Runtime - run time (String)
6. Genre - Array of Genre (Array of Strings)
7. Director - Array of Names of Directors (Array of Strings)
8. Writer - Array of Names of writers (Array of Strings)
9. Actors - Array of Names of writers (Array of Strings)
10. Plot - plot/ story (String)
11. Language - release language (String)
12. Country - Release country (String)
13. Awards - Array of award strings (Array of strings)
14. Poster - Link to movie poster (String)
15. Ratings - Array of ratings from different sources (Array of strings / json). Can be parsed out into separate columns.
16. Metascore - Metascore (Number)
17. imdbRating - IMDB rating (float)
18. imdbVotes - votes on imdb (Number)
19. imdbID - unique imdbID (String)
20. Type - Movie or something else i.e. OTT, music video etc (String)
21. DVD - DVD release date (Date)
22. Box office - revenue (String)

23. Production - Production house (String)
24. Website - Website (String)
25. Response - Response (String)

In [1]:
```
# importing all required libraries
import requests
import pandas as pd
```

In [47]:
```
# omdb api key
API_KEY = "fcd8bac6"
```

In [90]:
```
# Testing the API for one movie and it's release year
title = 'Avatar'
year = '2009'

# Calling the API and storing json response
movieInfo = requests.get('http://www.omdbapi.com/?apikey='+API_KEY+'&t='+title+
movieInfo
```

Out[90]:
```
{'Title': 'Avatar',
 'Year': '2009',
 'Rated': 'PG-13',
 'Released': '18 Dec 2009',
 'Runtime': '162 min',
 'Genre': 'Action, Adventure, Fantasy',
 'Director': 'James Cameron',
 'Writer': 'James Cameron',
 'Actors': 'Sam Worthington, Zoe Saldana, Sigourney Weaver',
 'Plot': 'A paraplegic Marine dispatched to the moon Pandora on a unique missi
on becomes torn between following his orders and protecting the world he feels
is his home.',
 'Language': 'English, Spanish',
 'Country': 'United States',
 'Awards': 'Won 3 Oscars. 89 wins & 131 nominations total',
 'Poster': 'https://m.media-amazon.com/images/M/MV5BZDA0OGQxNTItMDZkMC00N2UyLT
g3MzMtYTJmNjg3Nzk5MzRiXkEyXkFqcGdeQXVyMjUzOTY1NTc@._V1_SX300.jpg',
 'Ratings': [{'Source': 'Internet Movie Database', 'Value': '7.9/10'},
  {'Source': 'Rotten Tomatoes', 'Value': '82%'},
  {'Source': 'Metacritic', 'Value': '83/100'}],
 'Metascore': '83',
 'imdbRating': '7.9',
 'imdbVotes': '1,332,344',
 'imdbID': 'tt0499549',
 'Type': 'movie',
 'DVD': '22 Apr 2010',
 'BoxOffice': '$785,221,649',
 'Production': 'N/A',
 'Website': 'N/A',
 'Response': 'True'}
```

In [88]:
```
# For each imdb movie in it's top 250 list pull aditional details using above c
list = []

for index, movie in df_top_imdb_movies.iterrows():
    title = movie['title']
    year = movie['year']
    # Call omdb api for each movie and it's release year to get additional info
    movieInfo = requests.get('http://www.omdbapi.com/?apikey='+API_KEY+'&t='+ti
```

```
    # Append returned info in the list
    list.append(movieInfo)
```

In [89]:
```
# Read and store as a data frame
pd.set_option('display.max_columns', None)
df_omdb_movie_details = pd.DataFrame(list)
df_omdb_movie_details.head()
```

Out[89]:

	Title	Year	Rated	Released	Runtime	Genre	Director	Writer	Actors
0	The Shawshank Redemption	1994	R	14 Oct 1994	142 min	Drama	Frank Darabont	Stephen King, Frank Darabont	Tim Robbins, Morgan Freeman, Bob Gunton
1	The Godfather	1972	R	24 Mar 1972	175 min	Crime, Drama	Francis Ford Coppola	Mario Puzo, Francis Ford Coppola	Marlon Brando, Al Pacino, James Caan
2	The Dark Knight	2008	PG-13	18 Jul 2008	152 min	Action, Crime, Drama	Christopher Nolan	Jonathan Nolan, Christopher Nolan, David S. Goyer	Christian Bale, Heath Ledger, Aaron Eckhart
3	The Godfather Part II	1974	R	18 Dec 1974	202 min	Crime, Drama	Francis Ford Coppola	Francis Ford Coppola, Mario Puzo	Al Pacino, Robert De Niro, Robert Duvall
4	12 Angry Men	1957	Approved	10 Apr 1957	96 min	Crime, Drama	Sidney Lumet	Reginald Rose	Henry Fonda, Lee J. Cobb, Martin Balsam

Dataset 2: Netflix movies and shows

Dataset description:

About this Dataset: Netflix is one of the most popular media and video streaming platforms. They have over 8000 movies or tv shows available on their platform, as of mid-2021, they have over 200M Subscribers globally. This tabular dataset consists of listings of all the movies and tv shows available on Netflix, along with details such as - cast, directors, ratings, release year, duration, etc.

Data Source:

Kaggle https://www.kaggle.com/datasets/shivamb/netflix-shows

Metadata:

1. show_id: Unique ID for every Movie / Tv Show (String)
2. type: Identifier - A Movie or TV Show (String)
3. title: Title of the Movie / Tv Show (String)
4. director: Director Name (String)
5. cast: Actors involved in the movie / show [comma separated] (String)
6. country: Country where the movie / show was produced (String)
7. date_added: Date it was added on Netflix (Date)
8. release_year: Actual Release year of the move / show (Number)
9. rating: TV Rating of the movie / show (String)
10. duration: Total Duration - in minutes or number of seasons (String)
11. listed_in: Genre (String)
12. description: The summary description (String)

In [2]:
```
# Read the downloaded data file into pandas dataframe
df_netflix = pd.read_csv("netflix_titles.csv")
```

In [92]:
```
# check the data
df_netflix.head()
```

Out[92]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	d
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	

Dataset 3: Using webscraping to get movies and their ratings from IMDB website

Dataset description:

IMDB ranks movies and given them score. This dataset is published on IMDB website that includes top 250 movies of all time according to them.

Data Source:

https://www.imdb.com/chart/top

Metadata:

1. rank: Rank as given by IMDB (number)
2. title: Title or name of the movie (String)
3. rating: Rating or score given to the movie (Float)

4. year: Release year of the movie (Number)
5. cast: Actors names in the movie separated by columns (String)

In [18]:
```
# import all needed libraries
from requests import get
from bs4 import BeautifulSoup
from warnings import warn
from time import sleep
from random import randint
import numpy as np
import seaborn as sns
import re
```

In [77]:
```
# Downloading imdb top 250 movie's data while web scraping
url = 'http://www.imdb.com/chart/top'
response = requests.get(url)
soup = BeautifulSoup(response.text, "html.parser")
movies = soup.select('td.titleColumn')
crew = [a.attrs.get('title') for a in soup.select('td.titleColumn a')]
ratings = [b.attrs.get('data-value')
        for b in soup.select('td.posterColumn span[name=ir]')]



# create a empty list for storing
# movie information
list = []

# Iterating over movies to extract
# each movie's details
for index in range(0, len(movies)):

    # Separating movie into: 'place',
    # 'title', 'year'
    movie_string = movies[index].get_text()
    movie = (' '.join(movie_string.split()).replace('.', ''))
    movie_title = movie[len(str(index))+1:-7]
    year = re.search('\((.*?)\)', movie_string).group(1)
    place = movie[:len(str(index))-(len(movie))]
    data = {"rank": place,
            "title": movie_title,
            "rating": ratings[index],
            "year": year,
            "cast": crew[index],
            }
    list.append(data)


# Read and store as a data frame
df_top_imdb_movies = pd.DataFrame(list)
df_top_imdb_movies.head()
```

Out[77]:

	rank	title	rating	year	cast
0	1	The Shawshank Redemption	9.235818304767143	1994	Frank Darabont (dir.), Tim Robbins, Morgan Fre...
1	2	The Godfather	9.155851969250246	1972	Francis Ford Coppola (dir.), Marlon Brando, Al...
2	3	The Dark Knight	8.991872743468019	2008	Christopher Nolan (dir.), Christian Bale, Heat...
3	4	The Godfather Part II	8.98382721650489	1974	Francis Ford Coppola (dir.), Al Pacino, Robert...
4	5	12 Angry Men	8.95343842970773	1957	Sidney Lumet (dir.), Henry Fonda, Lee J. Cobb

Dataset 4: Get Monthly Top 100 Movies Torrents from Movie, TV, music search and download

Dataset Description:

Movie, TV, music search and download provides API connections to download torrent links for monthly top 100 movies, monthly top 100 tv shows, monthly top 100 music videos, monthly top 100 games torrent etc.

Data Source:

1. API link accessed through rapidapi - movie-tv-music-search-and-download.p.rapidapi.com

Metadata:

1. Title - Title or Name of the movie (String)
2. Torrent - Torrent link (String)
3. Size - Torrent size (String)
4. Rank - Torrent Rank

In [57]:
```
import requests

url = "https://movie-tv-music-search-and-download.p.rapidapi.com/monthly_top100

headers = {
        "X-RapidAPI-Key": "36a03a312bmshec916da8f40801cp12d965jsn00140a4bcf30",
        "X-RapidAPI-Host": "movie-tv-music-search-and-download.p.rapidapi.com"
}

response = requests.request("GET", url, headers=headers)
```

In [75]:
```
# import json
import json

# load API calls response into a json object
response_json = json.loads(response.text)

# instantiate an empty list
list = []

# loop through each json result and parse response json to get title, torrent,
# for each parsed result keep appending into a list
for data in response_json['result']:
    title = (data['title'].split('(')[0]).encode('utf-8').decode('utf-8')
    title = title.split('2022')[0]
    title = (title.split('2023')[0]).strip()
    torrent = data['torrent']
    size = data['size']

    data = {"title": title,
            "torrent_link": torrent,
            "size_of_torrent_file": size
    }

    list.append(data)
```

In [76]:
```
# Read into a data frame
df_movies_torrent = pd.DataFrame(list)
df_movies_torrent.head()
```

Out[76]:

	title	torrent_link	size_of_torrent_file
0	Avatar The Way of Water	https://itorrents.org/torrent/8BA89A34225442B0...	3.55 GB
1	Avatar The Way of Water	https://itorrents.org/torrent/2AA79E357A47CAD3...	1.73 GB
2	Creed III	https://itorrents.org/torrent/5E197437EC43CB71...	2.15 GB
3	Creed III	https://itorrents.org/torrent/12AF6C89A1189B10...	1.05 GB
4	Murder Mystery 2	https://itorrents.org/torrent/E38F0F91E9E7B170...	1.68 GB