Database Project Progress Report IMDB movies analysis

Professor- Nick Bear Brown

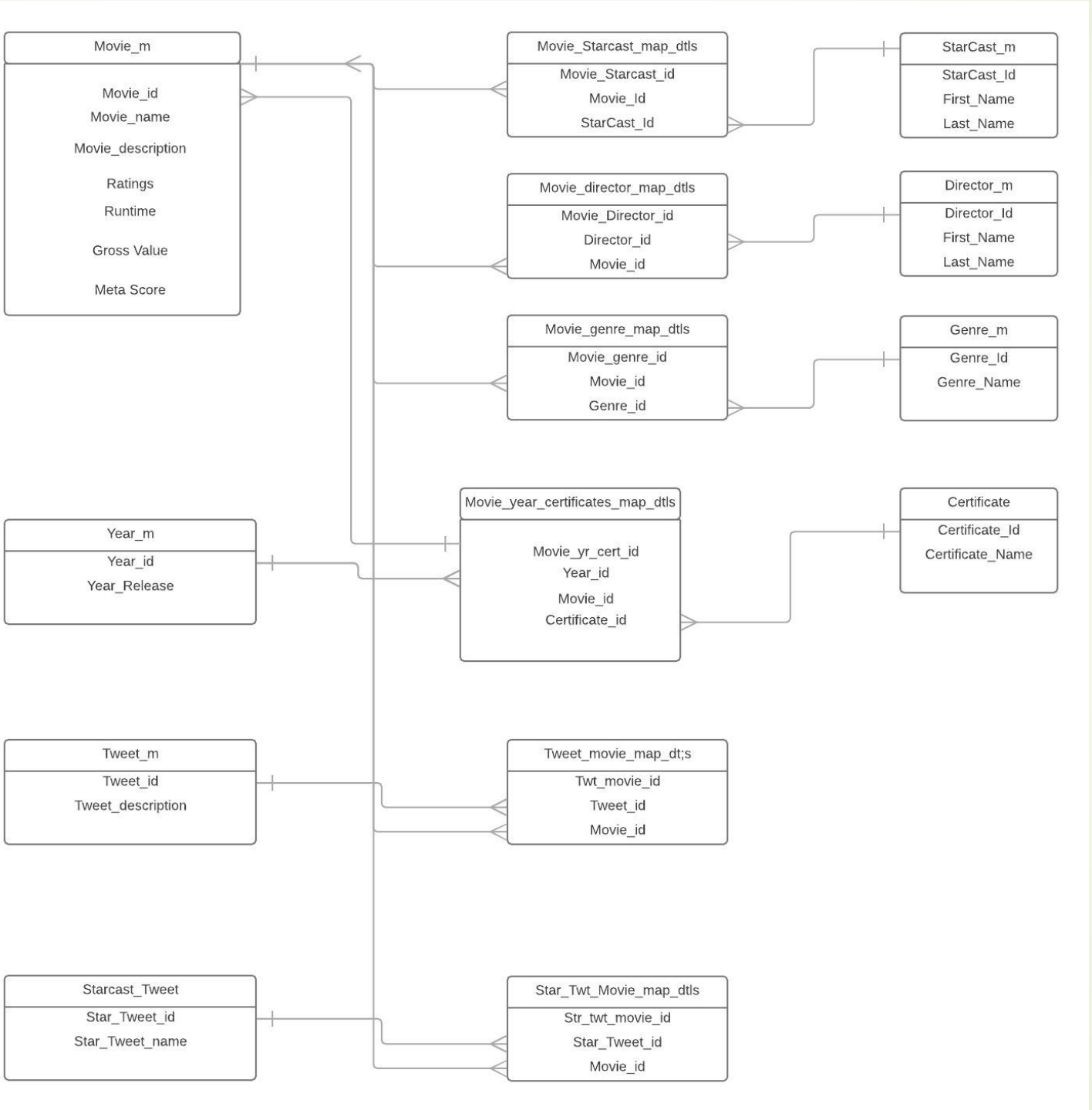
-By Ajay Goel

(001897443)

Project Division:

- Project is divided into 4 parts:
  - Web Scraping Imdb website.
  - Cleaning the data obtained from scraping.
  - Inserting the data into database
  - Searching the Movies on a social Network site

ER Diagram


Movie_m
Movie_id
Movie_name
Movie_description
Ratings
Runtime
Gross Value
Meta Score
Movie_Starcast_map_dtls
Movie_Starcast_id
Movie_Id
StarCast_Id
StarCast_m
StarCast_Id
First_Name
Last_Name
Movie_director_map_dtls
Movie_Director_id
Director_id
Movie_id
Director_m
Director_Id
First_Name
Last_Name
Movie_genre_map_dtls
Movie_genre_id
Movie_id
Genre_id
Genre_m
Genre_Id
Genre_Name
Year_m
Year_id
Year_Release
Movie_year_certificates_map_dtls
Movie_yr_cert_id
Year_id
Movie_id
Certificate_id
Certificate
Certificate_Id
Certificate_Name
Tweet_m
Tweet_id
Tweet_description
Tweet_movie_map_dt;s
Twt_movie_id
Tweet_id
Movie_id
Starcast_Tweet
Star_Tweet_id
Star_Tweet_name
Star_Twt_Movie_map_dtls
Str_twt_movie_id
Star_Tweet_id
Movie_id

1. Web Scraping the Imdb website

- Reference: http://www.imdb.com/search/title?release_date=2017&sort=num_votes,desc&page=1

- IMDB site for a while we find that while using advanced search feature we can have look at the best movies in the given time frame by 50 movies per page
- Check response first as if the website is not giving ideal response. ***If it is other than 200 than you are not able to fetch the data from the website.***
- While exploring the HTML content, it has two values "lister-item" and "mode - advanced". The combination is div containers.
- *First, extracted values are how many movies that page contains:*
- *Code Snippet:*

```
In [5]: id_check = html_soup.find(id ="main")
        movie_container = id_check.find_all(class_ ="lister-item mode-advanced")
        len(movie_container)

Out[5]: 50
```

- Now, analyzing the page has given various attributes like:
  - Movie_names
  - Year_release
  - Imdb_ratings
  - Metascores
  - Votes
  - Movie Description
  - Certificate
  - Runtime
  - Director Name
  - Star Cast
  - Gross Value

Extracting 1 page from Contaniers:

- Extracting all the values from the container.

```
#extract data from individual movie container
for container in movie_container:


    #if movie has Metascore, then extract:
    if container.find("div", class_ = "ratings-metascore") is not None:


        #the movie_name
        name = container.h3.a.text
        movie_names.append(name)

        #the year od release
        release = container.find("span", class_ = "lister-item-year text-muted unbold").text
        year_release.append(release)

        #the ratings for the movies
        ratings = float(container.strong.text)
        imdb_ratings.append(ratings)

        #the metascores
        meta = container.find("span", class_ = "metascore").text
        metascores.append(int(meta))

        #the votes
        vote = container.find("span", attrs = {"name":"nv"})['data-value']
        votes.append(int(vote))

        #the certificate
        certi = container.find("span", class_ = "certificate").text
        certificate.append(certi)
```

- Similarly, all other variables can be extracted:
- Conversion of the whole data in a Data Frame.

```
In [9]: test_df
```

Out[9]:

	certificate	director_name	genre	gross_value	imdb_ratings	metscores	movie_description	movie_names	runtime	star_cast	votes	year_release
0	R	James Mangold	\nAction, Drama, Sci-Fi	226,277,068	8.1	77	\nIn the near future, a weary Logan cares for ...	Logan	137 min	[James Mangold, Hugh Jackman, Patrick Stewart,...	460320	(2017)
1	PG-13	Patty Jenkins	\nAction, Adventure, Fantasy	412,563,408	7.5	76	\nWhen a pilot crashes and tells of conflict i...	Wonder Woman	141 min	[Patty Jenkins, Gal Gadot, Chris Pine, Robin W...	387198	(2017)
2	PG-13	Christopher Nolan	\nAction, Drama, History	188,373,161	8.0	94	\nAllied soldiers from Belgium, the British Em...	Dunkirk	106 min	[Christopher Nolan, Fionn Whitehead, Barry Keo...	365841	(2017)
3	PG-13	James Gunn	\nAction, Adventure, Sci-Fi	389,813,101	7.7	67	\nThe Guardians must fight to keep their newfo...	Guardians of the Galaxy Vol. 2	136 min	[James Gunn, Chris Pratt, Zoe Saldana, Dave Ba...	350810	(2017)
										[Rian Johnson,		

Script for Multiple Pages:

- Extracting top movies from 2010 – 2019.
- years = [str(i) for i in range(2010,2019)]
- Issue: Avoid flooding the server with tens of request per second, then we are much likely to avoid our IP being banned permanently.
- To avoid this:
  - Control the loop's rate by using the sleep() function in the python's "time" module. This will pause the execution of the loop for a specified amount of seconds.
  - T render our requests legit we will vary the amount of waiting time between requests by using the randint() function from python's "random" module.
- Monitoring the loop as it's still going
  - We have so many pages to scan through, to monitor them while we are looping through them.

- Code Snippet 1:

```
from IPython.core.display import clear_output
from time import time

#redeclaring the variables

movie_names = []
year_release = []
imdb_ratings = []
metascores = []
votes = []
movie_description = []
certificate = []
runtime = []
genre = []
director_name = []
star_cast = []
gross_value = []

#preparing the moniter of the loop
start_time = time()
requests = 0

#for every year in the interval 2000-2018
for year in years:

    #for every page in the onterval 1-4
    for page in pages:

        #make a get request
        response = get("http://www.imdb.com/search/title?release_date=" + year + "&sort=num_votes,desc&page=" + page)
```

- Code Snippet 2:

```
#the metascores
meta = container.find("span", class_ = "metascore").text
metascores.append(int(meta))

#the votes
vote = container.find("span", attrs = {"name":"nv"})['data-value']
votes.append(int(vote))

#the certificate
if container.find("span", class_ = "certificate") is not None:
    certi = container.find("span", class_ = "certificate").text
    certificate.append(certi)
else:
    certificate.append(None)

#the runtime
if container.find("span", class_ ="runtime") is not None:
    run = container.find("span", class_ ="runtime").text
    runtime.append(run)
else:
    runtime.append(None)

#the genre
if container.find("span", class_ ="genre") is not None:
    gen = container.find("span", class_ ="genre").text
    genre.append(gen)
else:
    genre.append(None)

#fetching all <p> tags
content = container.find_all("p")

#the description
```

- Code Snippet 3:

```
                #the director
                if content_2[0] is not None:
                    director = content_2[0].text
                    director_name.append(director)
                else:
                    director_name.append(None)

                #the gross value
                if len(container.find_all("span", attrs = {"name":"nv"})) >= 2:
                    gross = container.find_all("span", attrs = {"name":"nv"})[1]['data-value']
                    gross_value.append(gross)
                else:
                    gross_value.append(None)

                #extracting artists names
                if content_2[1] is not None:
                    temp = []
                    for i in range(len(content_2)-1):
                        temp.append(content_2[i].text)
                    star_cast.append(temp)
                else:
                    star_cast.append(None)
```

```
Request: 45, Frequency: 0.06508031892812906 requests/s
5 2018
```

- Here, we can see for 2018 year till 5th page our request is going.
- Total requests : 45

- Converting the Data Frame to excel so that it can be cleaned.

```
]: #storing scraped data into a data frame.

imdb_movie_dataset = pd.DataFrame({"movie_names":movie_names,
                                   "year_release":year_release,
                                   "imdb_ratings":imdb_ratings,
                                   "metscores":metascores,
                                   "votes":votes,
                                   "movie_description":movie_description,
                                   "certificate":certificate,
                                   "runtime":runtime,
                                   "genre":genre,
                                   "director_name": director_name,
                                   "star_cast": star_cast,
                                   "gross_value":gross_value
                                  })
```

```
]: imdb_movie_dataset.to_csv("imdb_movie_dataset.csv") #stores the Da
```

2 Part : Data Cleaning

- Observations :
- Various columns had garbage values like ^,!,\s,\n,[0-9]

```
imdb_movie_dataset = pd.read_csv("imdb_movie_dataset.csv") #reading the file we previously created.
import re

#the below code is used to clean the scraped data into a storable and further usable format.
#We have used regular expressions in order to do so

for i in range(len(imdb_movie_dataset['star_cast'])):

    #cleaning star_cast column
    imdb_movie_dataset['star_cast'][i] = re.findall(r"'([^']*)'",imdb_movie_dataset['star_cast'][i])

    #cleaning genre column
    imdb_movie_dataset['genre'][i] = re.findall("([^\\r\\n\s,][a-zA-Z]+)", imdb_movie_dataset['genre'][i])
```

- The updated data can be stored as a new csv.

2.1 Analyzing the Data:

- Checking the data types of the columns.
- Checking the null values in columns.
- Checking while the movie names have length = 0 as movie name's length cannot be 0.
- In Genre columns, data was in a list.
  - Splitting the data so that every movies genres can be in separate columns.
  - Reason: Data while dumping into SQL should be in First Normal Forms.
  - Similarly, in star cast column values were in a list.
  - Data after Cleaning:

```
]: Star_c1=Star_cast1['star_cast']
   Star_cast1['star_cast'] = Star_cast1.star_cast.apply(la
```

```
]: Star_c2 =Star_cast1['star_cast']
   Star_Split = Star_c2.str.split(',', expand=True)
   Star_Split
```

]:

	0	1	2	3
0	Christopher Nolan	Leonardo DiCaprio	Joseph Gordon-Levitt	Ellen Page

- Converting the data into data frames.
- Mapping all the columns according to the tables so that data is in 1 NF.
- Note: Data is cleaned and formed csv have no redundant data in one row.
- Code Snippet:

```
In [336]: Star_merge_data.to_csv("Star_merge_data.csv")

In [142]: Genre_DF.to_csv("Genrel.csv")

In [263]: Genre_merge_data.to_csv("Genre_merge_data.csv")

In [143]: movie = Data_file[['Movie_ID','movie_names','movie_description','runtime','imdb_ratings','gross_value','metscores']]

In [338]: year = Data_file[['Movie_ID','year_release']]
          director = Data_file[['Movie_ID','director_name']]
          Certificate = Data_file[['Movie_ID','certificate']]
          year.to_csv('year.csv',index=False)
          director.to_csv('director.csv',index=False)
          Certificate.to_csv('Certificate.csv',index=False)

In [145]: movie.to_csv('title.csv',index=False)
```

3. Inserting the data into Database using SQLite:

- Involves various steps:
  - Database connection.
  - Create table
    - Create table Function
    - Create table query
    - Insert Function
    - Insert Query
    - Reading data from csv

- Code Snippets:
- Create Query:

```
[93]: ## Create table queries
create_title_table = """ CREATE TABLE IF NOT EXISTS title_m (
                                    movie_id_pk INTEGER PRIMARY KEY,
                                    movie_id INTEGER,
                                    movie_names TEXT,
                                    movie_description TEXT,
                                    runtime TEXT,
                                    imdb_ratings TEXT,
                                    gross_value TEXT,
                                    metscores TEXT
                                    ); """
```

- Insert Query:

```
In [95]: def insert_title(cur, movie):
             """ insert into title table
             :param cur: cursor
             :param title_data: title data
             :return: current inserted row id
             """
             #movie['Movie_ID'],
             data = (movie['Movie_ID'],
                     movie['movie_names'],
                     movie['movie_description'],
                     movie['runtime'],
                     movie['imdb_ratings'],
                     movie['gross_value'],
                     movie['metscores'])
             sql="""INSERT INTO title_m('movie_id','movie_names','movie_description','runtime','imdb_ratings'
             ,'gross_value','metscores') VALUES(?,?,?,?,?,?,?)"""
             #hp_data=(certificate_data['certificate'])

             cur.execute(sql,data)

             return cur.lastrowid
```

- Making a connection and calling create and insert functions:

```
]: # create database
   ##res = create_database(database)
   # create a database connection
   cur = conn.cursor()
   df = pd.read_csv('/Users/ajaygoel/Documents/NEU/DBMS/PythonPrac/Project_DMDD/title_final.csv',encoding='latin-1');
   for index, row in df.iterrows():
       Movie_ID = insert_title(cur, row)

   conn.commit()
   conn.close()
```

- Similarly, all other tables are formed in the database.
- Please find the screenshots below:
- SQLite Snippet:

```
Ajays-MacBook-Pro:~ ajaygoel$ cd Documents/neu/dbms/PythonPrac
Ajays-MacBook-Pro:PythonPrac ajaygoel$ cd Project_DMDD/
Ajays-MacBook-Pro:Project_DMDD ajaygoel$ sqlite3
SQLite version 3.20.1 2017-08-24 16:21:36
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open Project_dbms.db
sqlite> .tables
certificate_m  genre_m        title_m
director_m     star_cast_m    year_m
sqlite>
```

- Data present in each table is in 1 NF.

- Selecting data from one table named as title_m :

```
sqlite> .mode column
sqlite> .headers on
sqlite> Select * from title_m;
movie_id_pk  movie_id    movie_names  movie_description

                                  runtime     imdb_ratings  gross_value  metscore
s
-----------  ----------  -----------  ----------------------------------------
-------------------------------------------------------------------
                                  ----------  ------------  -----------  --------
--
A thief, who steals corporate secrets through the use of dream-sharing technolog
y, is given the inverse task of planting an idea into the mind of a CEO.  148 mi
n     8.8           292576195.0  74
In 1954, a U.S. Marshal investigates the disappearance of a murderer, who escape
d from a hospital for the criminally insane.                              138 mi
n     8.1           128012934.0  63
The toys are mistakenly delivered to a day-care center instead of the attic righ
t before Andy leaves for college, and it's up to Woody to convince the o  103 mi
n     8.3           415004880.0  92
A committed dancer wins the lead role in a production of Tchaikovsky's "Swan Lak
e" only to find herself struggling to maintain her sanity.                108 mi
n     8.0           106954678.0  79
With the world now aware of his identity as Iron Man, Tony Stark must contend wi
th both his declining health and a vengeful mad man with ties to his fat  124 mi
n     7.0           312433331.0  57
The story of King George VI of the United Kingdom of Great Britain and Northern
Ireland, his impromptu ascension to the throne and the speech therapist   118 mi
n     8.0           138797449.0  88
A hapless young Viking who aspires to hunt dragons becomes the unlikely friend o
f a young dragon himself, and learns there may be more to the creatures   98 min
      8.1           217581231.0  74
Harvard student Mark Zuckerberg creates the social networking site that would be
come known as Facebook, but is later sued by two brothers who claimed he  120 mi
n     7.7           96962694.0   95
Dave Lizewski is an unnoticed high school student and comic book fan who one day
 decides to become a superhero, even though he has no powers, training o  117 mi
```

- Code Snippets of tables formed:
  - Director_m

```
sqlite> Select * from director_m;
director_id_pk  movie_id    director_name
--------------  ----------  ----------------
1               1           Christopher Nolan
2               2           Martin Scorsese
3               3           Lee Unkrich
4               4           Darren Aronofsky
5               5           Jon Favreau
6               6           Tom Hooper
7               7           Dean DeBlois
8               8           David Fincher
9               9           Matthew Vaughn
10              10          Pierre Coffin
11              11          David Yates
12              12          Tim Burton
13              13          Nathan Greno
14              14          Edgar Wright
15              15          Will Gluck
16              16          Danny Boyle
```

- Star_cast_m:

```
sqlite> Select * from star_cast_m;
star_cast_id_pk  movie_id    star1              star2              star3
           star4        star5       star6
---------------  ----------  -----------------  -----------------  -----------
---------  -----------  ----------  ----------
1                1           Christopher Nolan  Leonardo DiCaprio  Joseph Gord
on-Levitt   Ellen Page
2                2           Martin Scorsese    Leonardo DiCaprio  Emily Morti
mer         Mark Ruffa
3                3           Lee Unkrich        Tom Hanks          Tim Allen
           Joan Cusac
4                4           Darren Aronofsky   Natalie Portman    Mila Kunis
           Vincent Ca
5                5           Jon Favreau        Robert Downey Jr.  Mickey Rour
ke          Gwyneth Pa
```

- Genre_m

```
sqlite> Select * from genre_m;
genre_id_pk  movie_id    genre1      genre2      genre3
-----------  ----------  ----------  ----------  -------------------
1            1           Action       Adventure   Sci-Fi
2            2           Mystery      Thriller
3            3           Animation    Adventure   Comedy
4            4           Drama        Thriller
5            5           Action       Adventure   Sci-Fi
6            6           Biography    Drama
7            7           Animation    Action      Adventure
```

- Different tables formed as of now are :
  - Certificates_m
  - Genre_m
  - Title_m
  - Director_m
  - Star_cast_m
  - Year_m

4. Extracting Data from Social Media -Twitter

- In this, movies will be passed into twitter to extract tweets related to the movie.
- Movie and Tweets mapping will be created so that data can be analyzed further.
- Entities in the tweets will help us to answer various questions about the movies like:
  - How much movie is popular?
  - What are the similar tags used for expressing the movie?
  - How much influence does the posts have?

- First, the excel with the names of the movies will be uploaded.
  - We can also take the movies directly from the database by creating connection and running SQL query :
  - Code Snippet:

```
In [4]: rev = c.execute('Select tm.movie_names from title_m tm')
        rows = rev.fetchall()
        l=[]
        for row in rows:
            l.append(row[0])
            #print(row)
        print(l)
```

```
agon', 'The Social Network', 'Kick-Ass', 'Despicable Me', 'Ha
onderland', 'Tangled', 'Scott Pilgrim vs. the World', 'Easy A
les', 'Due Date', 'Tron', 'True Grit', 'Salt', 'RED', 'The Bo
e Sands of Time', 'Insidious', 'Robin Hood', 'The A-Team', 'T
s', 'The Tourist', 'Megamind', 'Predators', 'Machete', 'Unsto
hree Days', 'Love & Other Drugs', 'Get Him to the Greek', 'Pe
```

- Now, Space should not be there in movie names as when we search for any hashtag or names there is no space between the words.
- We will remove the spaces from the movie names.

```
In [42]: nospace = []
         for i in l:
             j = i.replace(' ','')
             nospace.append(j)

         print(nospace)
```

```
rd', 'Coco', 'TheMummy', "TheHitman'sBodyguard", 'Bright',
'Mother!', 'ValerianandtheCityofaThousandPlanets', 'Jumanji:
adyBird', 'Transformers:TheLastKnight', 'AmericanMade', 'Cal
Trainspotting', 'PowerRangers', 'TheBigSick', 'DarkestHour',
eturnofXanderCage', 'DespicableMe3', 'Annabelle:Creation',
rcle', 'WhatHappenedtoMonday', 'Okja', 'DeathNote', 'TheFore
Night', 'TheKillingofaSacredDeer', 'HappyDeathDay', 'ShotCal
ld'sGame", 'PhantomThread', 'Jigsaw', 'TheMountainBetweenUs'
oodTime', 'TheSpaceBetweenUs', 'TheSnowman', 'TheBeguiled',
```

Twitter Search:

- We will create various lists that we need from the tweets.
- We will access the Twitter by using it's API.
- Every user will have different authentication codes which will help in accessing the data:

In [44]:
```
import twitter

CONSUMER_KEY = 'gZRO12PVUncQN0QsNr1WHbYH4'
CONSUMER_SECRET = 't2wR8EbvBiYr23vGGaCUjE7gAJqH3rX2oTCEsxSKn1yEv410um'
OAUTH_TOKEN = '2974941042-wO8homuub4QlhSiU4LOdWR2fSjq2V1MTQ3UsxWs'
OAUTH_TOKEN_SECRET = 'zaubrsUNIBzsYWaVibCVTAQuAMUJxpXj9DMPEQ963Yifj'

auth = twitter.oauth.OAuth(OAUTH_TOKEN, OAUTH_TOKEN_SECRET,
                           CONSUMER_KEY, CONSUMER_SECRET)

twitter_api = twitter.Twitter(auth=auth)

# Nothing to see by displaying twitter_api except that it's now a
# defined variable

print(twitter_api)
```

```
<twitter.api.Twitter object at 0x10dae8f60>
```

- Entities we will extract from twitter are:
  - Hashtags
  - Locations of the user
  - Text in the tweet
  - Favorite count of the tweet
  - Language of the tweet
  - Place of the user
  - Geography of the user
  - User Description
  - User Followers Count
  - User Time Zone
  - Whether we are following or not

- Issue: The API rate limits described for Twitter is only 180 requests per 15 minutes per user.
- We cannot send more than 180 requests in 15 minutes. For this, we can use sleep function in python.
- It will help the code to sleep as soon as it reaches limit.

```
In [50]: import time as time# import sleep
```

- Kept one counter which will count the requests as movie count.
- As soon as it reached count 170, code will halt and will rest restart after 900 seconds.
- Movies names will be printed which have been passed in the Twitter.
- It will help to analyze whether the process is running or not.

- Code Snippets 1:

```
import json
from urllib.parse import unquote

moviecount=0

for movie in title:
    moviecount+=1

    n=199

    search_results = twitter_api.search.tweets(q=movie, count=n)
    statuses = search_results['statuses']

    hashtags = [hashtag['text']
    for status in statuses
        for hashtag in status['entities']['hashtags']]
    #print(hashtags)

    s1=len(statuses)#.size
    for q in range(s1):
        l1.append(statuses[q]['user']['location'])
```

- Code Snippet 2:

```
locations = l1
retweets = [
        (status['text'],
         status['favorite_count'],
         status['lang'],
         status['place'],
         status['geo'],
         status['user']['description'],
         status['user']['followers_count'],
         status['user']['time_zone'])
        for status in statuses
       ]

for countHash in hashtags:
    moviesTest1.append(movie)
    HashtagsTest1.append(hashtags)

for countLoc in locations:
    moviesTest12.append(movie)
    l2.append(countLoc)

for countWholeData in retweets:
    moviesTest13.append(movie)
    whole_tweet_data.append(countWholeData)
```

- Code Snippet 3:

```
if(moviecount%170==0):
    time.sleep(900)
print(movie)
```

- If the movies count is not 170 then it will print the movie which is passed.
- Now, we have data frames separately of different entities like:
  - Hashtags
  - Locations of the user
  - Text in the tweet
  - Favorite count of the tweet
  - Language of the tweet
  - Place of the user
  - Geography of the user
  - User Description
  - User Followers Count
  - User Time Zone
  - Whether we are following or not

- We can map movies also so that there can be various separate tables like
  - Users Data and movies mapping: It will help us in answering various questions related to the movies.
  - Code Snippet:

```
Tweet_text = df11[0]
Tweet_fav_count = df11[1]
Tweet_language = df11[2]
Tweet_place = df11[3]
Tweet_geography = df11[4]
Tweet_user_description= df11[5]
Tweet_user_follower_count = df11[6]
Tweet_user_time_zone = df11[7]

Tweet_Data = pd.DataFrame({'Movies':moviesTest13,'Tweet_text':Tweet_text,'Tweet_fav_
                           :Tweet_language,'Tweet_place':Tweet_place,'Tweet_geograph
                           'Tweet_user_description':Tweet_user_description,'Tweet_us
                           :Tweet_user_follower_count,'Tweet_user_time_zone':Tweet_u
```

- Data Frame formed:

```
In [95]: Tweet_Data
```

Movies	Tweet_fav_count	Tweet_geography	Tweet_language	Tweet_place	Tweet_text	Tweet_user_description	Tweet_user_follower_count	Tv
Inception	0	None	en	None	Since its inception over 20 years ago, the Civ...	Nous formons principalement des apprentis méca...	986	
Inception	0	None	en	None	I watched inception and the British shape shif...	ahhhhhhhhhhhh	153	
Inception	0	None	en	None	@BogiGroB @LorrieAnneRider @JGWelcometomex @Sw...	Singer, Writer, Feminist. A varied mix of in...	482	

- We can create csv and also dump it in the database following the previous process by creating connecting, writing create query with all the columns and insert function.

- Create and Insert:

```
create_user_table = """ CREATE TABLE IF NOT EXISTS user_tweet_movies_m (
                                    user_tweet_mov_id INTEGER PRIMARY KEY,
                                    movies TEXT,
                                    tweet_text TEXT,
                                    tweet_fav_count INTEGER,
                                    tweet_language TEXT,
                                    tweet_place TEXT,
                                    tweet_geography TEXT,
                                    tweet_user_description TEXT,
                                    tweet_user_follower_count INTEGER,
                                    tweet_user_time_zone TEXT
                                    ); """
```

```
def insert_user(cur, user):
    """ insert into user table
    :param cur: cursor
    :param title_data: title data
    :return: current inserted row id
    """
    #movie['Movie_ID'],
    data = (user['Movies'],
            user['Tweet_text'],
            user['Tweet_fav_count'],
            user['Tweet_language'],
            user['Tweet_place'],
            user['Tweet_geography'],
            user['Tweet_user_description'],
            user['Tweet_user_follower_count'],
            user['Tweet_user_time_zone'])
    sql="""INSERT INTO user_tweet_movies_m('movies','tweet_text','tweet_fav_count','tweet_
    ,'tweet_geography','tweet_user_description','tweet_user_follower_count','tweet_user_ti
    #hp_data=(certificate_data['certificate'])
```

- Counting data in the table:

```
sqlite> Select count(*) from user_tweet_movies_m;
103516
sqlite>
```

- The table user_tweet_movies_m has 103516 data which is extracted from twitter.

- Questions:
- i. What are people saying about me (somebody)?

```
In [7]: rev = c.execute('Select um.movies,um.tweet_text from title_m tm,user_tweet_movies_m um where trim(tm.movie_names)=trim(

In [8]: rows = rev.fetchall()
        rows

Out[8]: [('Inception',
          'Since its inception over 20 years ago, the Civic Type R has delivered the highest possible performing version of t
        h… https://t.co/rN9EN5Bv1G'),
         ('Inception',
          'I watched inception and the British shape shifting guy was really hot hddisnxksbfhdks https://t.co/KOhwkgM0l1'),
         ('Inception',
          '@BogiGroB @LorrieAnneRider @JGWelcometomex @Swisssusan @DJ4JG @claireangel2 @NeshkaVirdure @Glorygirlone @PTeach11
        1… https://t.co/6WcuqlXn2k'),
         ('Inception',
          '@ManagerEnrique @rogerfaulkner94 For what? The Cincy Saints? Fake news, mate. FCC was an MLS vehicle from the mome
```

- We can select texts of the tweets related to the movie. This will answer that what the people are saying about the movie.

- ii. How viral are my posts?
- The number of retweets on the tweet can give us this answer.

Count	Screen Name	Tweet ID	Text
186	sebtsb	974038800048500736	RT @sebtsb: IF U LIVE IN EUROPE/AUSTRALI/ASIA; FINDING ME IS PROBABLY ALREADY OUT 💛 START STREAMING IT NOW! LETS SEE HOW MANY PLAYS WE CA…
54	YongSunpuff	974485462533353474	RT @YongSunpuff: GOD. HOW. DO. I. CONTINUE. TO. LIVE. NOW. https://t.co/kVyZICHwgn
31	5Strat	974108568021450752	RT @5Strat: When's #NationalWalkoutDay to end

- iii. How much influence to my posts have?
  - Favorites count of the tweet can give this answer.
- v. What users post like me?
  - Sentimental analysis positive, negative and neutral result will help us in giving the details. If it is positive then posts are similar. If not then they are not similar.
- vi. Who should I be following?
  - Sentimental analysis positive, negative and neutral result will help us in giving the details. If Posts are similar and I am not following then I can follow user.
- vii. What topics are trending in my domain?
  - I will search for the reply in similar tweets. If the count of the reply is more than that topic is more trending
- iv. What posts are like mine?
  - We can do sentimental analysis on the basis of tweets. If the analysis is positive then it can help us in answering this question.
- What keywords/ hashtags should I add to my post?
  - Analysis of similar tweets can help in adding hashtags which are not in mine tweets.

- Should I follow somebody back?
  - Similar Tweets and if not following then we can follow back.
- x. What is the best time to post?
  - Grouping tweet time can help us in analyzing this.

```
In [13]: rev = c.execute('Select um.movies,um.tweet_fav_count from user_tweet_movies_m um group by um.movies')

In [14]: rows2 = rev.fetchall()
         rows2

Out[14]: [(None, None),
          ('#Selfmde young stunner..still chasing the drm #BIAFRA', 1422),
          ('1008', 'Pacific Time (US & Canada)'),
          ('12 Years a Slave', 0),
          ('127 Hours', 0),
          ('129', 'Pacific Time (US & Canada)'),
          ('1291', 'London'),
          ('13', 0),
          ('188', None)
```

- xi. Should I add and picture or url to my post?
  - I will check the retweet count of the tweet with the url present in the tweet. If it is more than the retweet count of the normal tweets with no url then I will add url otherwise not.
- xiii. What's my reach?
  - I will try to answer this by making count of the common words of the movies, that will help in calculating what words are in my reach.