Movie Recommender system

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import ast
```

Reading the dataset

In [73]:
```
movies = pd.read_csv(r"G:\rec_sys\movies_metadata.csv")
credits = pd.read_csv(r"G:\rec_sys\credits.csv")
links = pd.read_csv(r"G:\rec_sys\links.csv")
```

```
C:\Users\lokit\AppData\Local\Temp\ipykernel_26964\798319844.py:1: DtypeWarni
ng: Columns (10) have mixed types. Specify dtype option on import or set low
_memory=False.
  movies = pd.read_csv(r"G:\rec_sys\movies_metadata.csv")
```

In [5]:
```
#Checking the datatype of links dataset
links.dtypes
```

Out[5]:
```
movieId      int64
imdbId       int64
tmdbId     float64
dtype: object
```

Cleaning dataset

In [6]:
```
#In the movies dataset,the ID column contains[] so replacing with 0
movies['id'].replace('[]',0,inplace=True)
```

In [7]:
```
#Changing the datatype to numeric
movies['id'] = pd.to_numeric(movies['id'])
```

In [8]:
```
#Merging movies and credits dataset with the common column 'id'
movies_credits = pd.merge(movies,credits,on='id')
```

In [9]:
```
movies_credits.head(2)
```

Out[9]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdl
0	FALSE	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt011
1	FALSE	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	tt011

2 rows × 26 columns

In [10]:
```
#Displaying first 3 rows of links dataset
links.head(3)
```

Out[10]:

	movieId	imdbId	tmdbId
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0

In [11]:
```
#Finding the datatypes of links
links.dtypes
```

Out[11]:
```
movieId      int64
imdbId       int64
tmdbId     float64
dtype: object
```

In [12]:
```
#Checking the null values of links dataset
links.isnull().sum()
```

Out[12]:
```
movieId      0
imdbId       0
tmdbId     219
dtype: int64
```

In [13]:
```
#As on the links dataset 'tmdbId' is replaced with 'id' (because all the id's
links.rename(columns={'tmdbId':'id'},inplace=True)
```

In [14]:
```
#Dropping the 'imdb_id' column
movies_credits = movies_credits.dropna(axis=0, subset=['imdb_id'])
```

In [15]:
```
#On 'imdb_id' we can see the values in 'tt' so lets remove it
movies_credits['imdb_id'] = movies_credits['imdb_id'].str.replace('tt''[0-7]'
```

In [16]:
```
#Changing the datatype
movies_credits['imdb_id'] = pd.to_numeric(movies_credits['imdb_id'])
```

In [17]:
```
#Rename the columns
links.rename(columns={'imdbId':'imdb_id'},inplace=True)
```

In [18]:
```
links
```

Out[18]:

	movieId	imdb_id	id
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0
3	4	114885	31357.0
4	5	113041	11862.0
...	...	...	...
45838	176269	6209470	439050.0
45839	176271	2028550	111109.0
45840	176273	303758	67758.0
45841	176275	8536	227506.0
45842	176279	6980792	461257.0

45843 rows × 3 columns

In [19]:
```
links1 = links[['movieId','imdb_id']]
```

In [20]:
```
#Merging movies_credits and links1 with the common 'imdb_id' columns
movies_credits = pd.merge(movies_credits,links1,on='imdb_id')
```

In [21]:
```
#Displaying first 2 rows
movies_credits.head(2)
```

Out[21]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb
0	FALSE	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	1147
1	FALSE	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	1134

2 rows × 27 columns

In [22]:
```
#Reading the dataset
keywords = pd.read_csv(r"G:\rec_sys\keywords.csv")
```

In [23]:
```
#Displaying first 2 rows
keywords.head(2)
```

Out[23]:

	id	keywords
0	862	[{'id': 931, 'name': 'jealousy'}, {'id': 4290,...
1	8844	[{'id': 10090, 'name': 'board game'}, {'id': 1...

In [24]:
```
#Checking the datatypes
keywords.dtypes
```

Out[24]:
```
id           int64
keywords    object
dtype: object
```

In [25]:
```
#Merging movies_credits and keywords dataset based on 'id' column
movies_credits = pd.merge(movies_credits,keywords,on='id')
```

In [26]:
```
#Displaying first 2 rows
movies_credits.head(2)
```

Out[26]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb
0	FALSE	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	1147
1	FALSE	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	1134

2 rows × 28 columns

In [27]:
```
#As in Genre columns we found all the unwanted data so we scrapped the genre s
def genre_name(obj):
    L = []
    for i in ast.literal_eval(obj):
        L.append(i['name'])
    return L
```

In [28]:
```
#Applying the function 'genre_name' on 'genres' column
movies_credits['genres'] = movies_credits['genres'].apply(genre_name)
```

In [29]:
```
#Displaying first 2 rows
movies_credits.head(2)
```

Out[29]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_
0	FALSE	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[Animation, Comedy, Family]	http://toystory.disney.com/toy-story	862	1147(
1	FALSE	NaN	65000000	[Adventure, Fantasy, Family]	NaN	8844	1134(

2 rows × 28 columns

In [30]:
```
#All the cast name was under 'name' section so scrapped the data and stored o
def cast_name(obj):
    L = []
    counter = 0
    for i in ast.literal_eval(obj):
        if counter != 3:
            L.append(i['name'])
            counter+=1
        else:
            break
    return L
```

In [31]:
```
#Applying the function 'cast_name' on 'cast' column
movies_credits['cast'] = movies_credits['cast'].apply(cast_name)
```

In [32]:
```
movies_credits.head(2)
```

Out[32]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_
0	FALSE	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[Animation, Comedy, Family]	http://toystory.disney.com/toy-story	862	1147(
1	FALSE	NaN	65000000	[Adventure, Fantasy, Family]	NaN	8844	1134(

2 rows × 28 columns

In [33]:
```
#Director name is scrapped from 'crew' column
def director_name(obj):
    L = []
    for i in ast.literal_eval(obj):
        if i['job'] == 'Director':
            L.append(i['name'])
            break
    return L
```

In [34]:
```
#Applying the function 'director_name' on 'crew' column
movies_credits['crew'] = movies_credits['crew'].apply(director_name)
```

In [35]:
```
#Applying the function 'genre_name' on 'keywords' column
movies_credits['keywords'] = movies_credits['keywords'].apply(genre_name)
```

In [36]:
```
#Displaying first 2 rows
movies_credits.head(2)
```

Out[36]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_
0	FALSE	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[Animation, Comedy, Family]	http://toystory.disney.com/toy-story	862	1147(
1	FALSE	NaN	65000000	[Adventure, Fantasy, Family]	NaN	8844	1134(

2 rows × 28 columns

In [37]:
```
#We can see that there are integer values present in the language column.
#Let's take a look at it.
#Dropping the specific rows mentioned below
lang_drop = movies_credits[(movies_credits['original_language'] == '82.0')

movies_credits.drop(index = lang_drop, inplace = True)
```

In [38]:
```
#Checking the null values
movies_credits['original_language'].isnull().sum()
```

Out[38]: 9

In [39]:
```
movies_credits.dropna(subset = ['original_language'], inplace = True)# droppi
```

In [40]:
```
lang_dec = {'en' : 'English', 'fr' : 'French', 'zh' : 'Chinese', 'it' : 'Ital
                    'nl' : 'Dutch', 'de' : 'German', 'cn' : 'Chinese','ar' : '
                    'ru' : 'Russian', 'sv' : 'Swedish', 'ja' : 'Japanese', 'ko
                    'bn' : 'Bengali', 'he' : 'Hebrew', 'pt' : 'Portuguese', 'w
                    'hu' : 'Hungarian', 'cy' : 'Welsh', 'vi' : 'Vietnamese', '
                    'no' : 'Norwegian', 'nb' : 'Norwegian', 'pl' : 'Polish', '
                    'mk' : 'Macedonian', 'bo' : 'Tibetan', 'ca' : 'Catalan', '
                    'sk' : 'Slovak', 'bs' : 'Bosnian', 'hi' : 'Hindi', 'tr' :
                    'ps' : 'Pashto', 'ab' : 'Abkhazian','eo' : 'Esperanto','ka
                    'bm' : 'Bambara', 'zu' : 'Zulu', 'uk' : 'Ukrainian', 'af'
                    'et' : 'Estonian', 'ku' : 'Kurdish', 'fy' : 'Frisian', 'lv
                    'sl' : 'Slovenian', 'tl' : 'Tagalog', 'ur' : 'Urdu', 'rw'
                    'bg' : 'Bulgarian', 'mr' : 'Marathi', 'lt' : 'Lithuanian',
                    'sq' : 'Albanian', 'qu' : 'Quechua', 'te' : 'Telugu', 'am'
                    'tg' : 'Tajik', 'ml' : 'Malayalam','hr' : 'Croatian', 'lo'
                    'kn' : 'Kannada', 'eu' : 'Basque', 'ne' : 'Nepali', 'pa' :
                    'gl' : 'Galician', 'uz' : 'Uzbek', 'sm' : 'Samoan', 'mt' :
                    'iu' : 'Inuktitut', 'lb' : 'Luxembourgish', 'si' : 'Sinhal

}


movies_credits['original_language'] = movies_credits['original_language'].map
```

In [41]:
```
movies_credits['release_date'].isnull().sum()#Checking the null values
```

Out[41]:
```
33
```

In [42]:
```
#Dropping the null values from 'release_date' column
movies_credits.dropna(subset = ['release_date'], inplace = True)
```

In [43]:
```
#Applying 'release_date' into datetime format
movies_credits['release_date'] = movies_credits['release_date'].apply(pd.to_d
```

```
C:\Users\lokit\anaconda3\anaconda\lib\site-packages\pandas\core\apply.py:
1137: UserWarning: Parsing '30-10-1995' in DD/MM/YYYY format. Provide for
mat or specify infer_datetime_format=True for consistent parsing.
  mapped = lib.map_infer(
C:\Users\lokit\anaconda3\anaconda\lib\site-packages\pandas\core\apply.py:
1137: UserWarning: Parsing '15-12-1995' in DD/MM/YYYY format. Provide for
mat or specify infer_datetime_format=True for consistent parsing.
  mapped = lib.map_infer(
C:\Users\lokit\anaconda3\anaconda\lib\site-packages\pandas\core\apply.py:
1137: UserWarning: Parsing '22-12-1995' in DD/MM/YYYY format. Provide for
mat or specify infer_datetime_format=True for consistent parsing.
  mapped = lib.map_infer(
C:\Users\lokit\anaconda3\anaconda\lib\site-packages\pandas\core\apply.py:
1137: UserWarning: Parsing '16-11-1995' in DD/MM/YYYY format. Provide for
mat or specify infer_datetime_format=True for consistent parsing.
  mapped = lib.map_infer(
C:\Users\lokit\anaconda3\anaconda\lib\site-packages\pandas\core\apply.py:
1137: UserWarning: Parsing '17-11-1995' in DD/MM/YYYY format. Provide for
mat or specify infer_datetime_format=True for consistent parsing.
```

In [44]:
```
#Fetching the year from 'release_year' columns
movies_credits['release_year']  = movies_credits['release_date'].apply(lambda
```

In [45]:
```
#Displaying the 'release_year' column
movies_credits['release_year']
```

Out[45]:
```
0        1995
1        1995
2        1995
3        1995
4        1995
         ...
31082    2000
31083    1995
31084    1991
31085    2003
31086    1917
Name: release_year, Length: 31045, dtype: int64
```

In [46]:
```
#Fetching the data,in which the runtime is less than 15
movies_credits[movies_credits['runtime'] <= 15].shape
```

Out[46]:
```
(1469, 29)
```

In [47]:
```
median_runtime = movies_credits['runtime'].median() # getting the median valu
movies_credits['runtime'] = movies_credits['runtime'].replace(list(range(0, 1
```

In [48]:
```
movies_credits['runtime'].isnull().sum()#Checking the null values runtime col
```

Out[48]:
```
143
```

In [49]:
```
movies_credits.dropna(subset = ['runtime'], inplace = True)#Dropping the rows
```

In [50]:
```
movies_credits.isnull().sum()#Checking the null values from movies_credits da
```

Out[50]:
```
adult                        0
belongs_to_collection    27606
budget                       0
genres                       0
homepage                 28333
id                           0
imdb_id                      0
original_language           26
original_title               0
overview                   454
popularity                   0
poster_path                204
production_companies         0
production_countries         0
release_date                 0
revenue                      0
runtime                      0
spoken_languages             0
status                      50
tagline                  16193
title                        0
video                        0
vote_average                 0
vote_count                   0
cast                         0
crew                         0
movieId                      0
keywords                     0
release_year                 0
dtype: int64
```

In [51]:
```
#Applying the function 'genre_name' on 'production_companies' column
movies_credits['production_companies'] = movies_credits['production_companies
```

In [92]:
```
#Applying the function 'genre_name' on 'production_countries' column
movies_credits['production_countries'] = movies_credits['production_countries
```

In [1]:
```
from statistics import mode
```

In [53]:
```
#Filling nan with mode values
movies_credits['status'].fillna(movies_credits['status'].mode()[0],inplace=Tr
```

In [54]:
```
#Displaying
movies_credits['status']
```

Out[54]:
```
0        Released
1        Released
2        Released
3        Released
4        Released
          ...
31082    Released
31083    Released
31084    Released
31085    Released
31086    Released
Name: status, Length: 30902, dtype: object
```

In [55]:
```
#Filling nan values with mode values
movies_credits['original_language'].fillna(movies_credits['original_language'
```

In [56]:
```
#Dispalying
movies_credits['original_language']
```

Out[56]:
```
0        English
1        English
2        English
3        English
4        English
          ...
31082    English
31083    English
31084    English
31085    English
31086    English
Name: original_language, Length: 30902, dtype: object
```

In [57]:
```
#Reading the dataset
ratings = pd.read_csv(r"G:\rec_sys\ratings_small.csv")
```

In [58]:
```
#Displaying
ratings.head()
```

Out[58]:

	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182
3	1	1129	2.0	1260759185
4	1	1172	4.0	1260759205

In [59]:
```
#Dropping the timestamp
ratings.drop('timestamp',axis=1,inplace=True)
```

In [60]:
```
ratings
```

Out[60]:

	userId	movieId	rating
0	1	31	2.5
1	1	1029	3.0
2	1	1061	3.0
3	1	1129	2.0
4	1	1172	4.0
...	...	...	...
99999	671	6268	2.5
100000	671	6269	4.0
100001	671	6365	4.0
100002	671	6385	2.5
100003	671	6565	3.5

100004 rows × 3 columns

In [61]:
```
ratings = ratings[['movieId','rating']]
```

In [62]:
```
#Applying the groupby operation on 'movieId'
ratings1 = pd.DataFrame(ratings.groupby('movieId')['rating'].mean())
```

In [71]:
```
#Displaying
movies1 = pd.DataFrame(movies_credits[['title','movieId','release_year']])
```

In [65]:
```
#Merging the dataset 'movies1' and 'ratins1' with common column 'movieId'
ratings2 = pd.merge(movies1,ratings1,on='movieId')
```

In [66]:
```
#Displaying
ratings2
```

Out[66]:

	title	movieId	release_year	rating
0	Toy Story	1	1995	3.872470
1	Jumanji	2	1995	3.401869
2	Grumpier Old Men	3	1995	3.161017
3	Waiting to Exhale	4	1995	2.384615
4	Father of the Bride Part II	5	1995	3.267857
...	...	...	...	...
7660	Michael Jackson's Thriller	152173	1983	3.000000
7661	Demons	152844	1971	4.000000
7662	The Video Dead	159462	1987	3.000000
7663	The Legend of Tarzan	160563	2016	2.500000
7664	The Last Brickmaker in America	161944	2001	5.000000

7665 rows × 4 columns

In [74]:
```
#Replacing the column name
movies_credits['imdbid'] = movies['imdb_id']
```

In [94]:
```
#Displaying
movies_credits['production_countries'].head()
```

Out[94]:
```
0    [United States of America]
1    [United States of America]
2    [United States of America]
3    [United States of America]
4    [United States of America]
Name: production_countries, dtype: object
```

In [80]:
```
#Checking the datatype
movies_credits.dtypes
```

Out[80]:
```
adult                          object
belongs_to_collection          object
budget                          int64
genres                         object
homepage                       object
id                              int64
imdb_id                         int64
original_language              object
original_title                 object
overview                       object
popularity                    float64
poster_path                    object
production_companies           object
production_countries           object
release_date           datetime64[ns]
revenue                       float64
runtime                       float64
spoken_languages               object
status                         object
tagline                        object
title                          object
video                          object
vote_average                  float64
vote_count                    float64
cast                           object
crew                           object
movieId                         int64
keywords                       object
release_year                    int64
imdbid                         object
dtype: object
```

In [77]:
```
#Changing the datatype
movies_credits['popularity'] = pd.to_numeric(movies_credits['popularity'])
```

In [79]:
```
#Changing the datatype
movies_credits['budget'] = pd.to_numeric(movies_credits['budget'])
```

In [81]:
```
#Checking the null values from  the respective columns
movies_credits.isnull().sum()
```

Out[81]:
```
adult                         0
belongs_to_collection     27606
budget                        0
genres                        0
homepage                  28333
id                            0
imdb_id                       0
original_language             0
original_title                0
overview                    454
popularity                    0
poster_path                 204
production_companies          0
production_countries          0
release_date                  0
revenue                       0
runtime                       0
spoken_languages              0
status                        0
tagline                   16193
title                         0
video                         0
vote_average                  0
vote_count                    0
cast                          0
crew                          0
movieId                       0
keywords                      0
release_year                  0
imdbid                       12
dtype: int64
```

In [91]:
```
#Fetching the data by using loc opeartion
movies_credits.loc[movies_credits['original_title']!=movies_credits['title']]
```

Out[91]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_lar
29	FALSE	NaN	18000000	[Fantasy, Science Fiction, Adventure]	NaN	902	112682	
30	FALSE	NaN	0	[Drama, Crime]	NaN	37557	115012	C
33	FALSE	NaN	0	[Romance, Adventure]	NaN	78802	114952	
58	FALSE	NaN	0	[Comedy, Drama, Romance]	NaN	11010	110877	
59	FALSE	NaN	0	[Thriller, Drama, Mystery]	NaN	99040	112714	

5 rows × 30 columns

In [95]:
```
#Dropping the unwanted columns
movies_credits.drop(['belongs_to_collection','homepage','adult','video','orig
```

In [96]:
```
##flatten list,join all tags overview etc
```

In [98]:
```
#Replacement
movies_credits['genres'] = movies_credits['genres'].apply(lambda x:[i.replace
movies_credits['keywords'] = movies_credits['keywords'].apply(lambda x:[i.rep
movies_credits['cast'] = movies_credits['cast'].apply(lambda x:[i.replace(' '
movies_credits['crew'] = movies_credits['crew'].apply(lambda x:[i.replace(' '
```

In [107]:
```
#Changing the datatype
movies_credits['overview'] = movies_credits['overview'].astype(str)
```

In [108]:
```
movies_credits['overview'] = movies_credits['overview'].apply(lambda x:x.spli
```

In [122]:
```
#Creating a feature column in which it carries multiple column('overview','ge
movies_credits['features'] = movies_credits['overview']+movies_credits['genre
```

In [118]:
```
movies_credits.columns
```

Out[118]:
```
Index(['budget', 'genres', 'id', 'imdb_id', 'original_language', 'overview',
       'popularity', 'poster_path', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime', 'taglin
e',
       'title', 'vote_average', 'vote_count', 'cast', 'crew', 'movieId',
       'keywords', 'release_year', 'imdbid'],
      dtype='object')
```

In [117]:
```
#Dropping the 'spoken_language' column
movies_credits.drop(columns='spoken_languages',axis=1,inplace=True)
```

Replacement and joining opeartion

In [148]:
```
movies_credits['features'].apply(lambda x:[i.replace(',','') for i in x ])
```

Out[148]:
```
0        [Led, by, Woody, Andy's, toys, live, happily, ...
1        [When, siblings, Judy, and, Peter, discover, a...
2        [A, family, wedding, reignites, the, ancient, ...
3        [Cheated, on, mistreated, and, stepped, on, th...
4        [Just, when, George, Banks, has, recovered, fr...
                              ...
31082    [A, film, archivist, revisits, the, story, of,...
31083    [It's, the, year, 3000, AD., The, world's, mos...
31084    [Yet, another, version, of, the, classic, epic...
31085    [When, one, of, her, hits, goes, wrong, a, pro...
31086    [In, a, small, town, live, two, brothers, one,...
Name: features, Length: 30902, dtype: object
```

In [153]:
```
movies_credits['features'] = movies_credits['features'].apply(lambda x:' '.jo
```

In [155]:
```
movies_credits['keywords'].apply(lambda x:' '.join(x))
```

Out[155]:
```
0        jealousy toy boy friendship friends rivalry bo...
1        boardgame disappearance basedonchildren'sbook ...
2           fishing bestfriend duringcreditsstinger oldmen
3        basedonnovel interracialrelationship singlemot...
4        baby midlifecrisis confidence aging daughter m...
                               ...
31082     witch mythology legend serialkiller mockumentary
31083
31084
31085
31086
Name: keywords, Length: 30902, dtype: object
```

In [157]:
```
movies_credits['cast'] = movies_credits['cast'].apply(lambda x:' '.join(x))
movies_credits['crew'] = movies_credits['crew'].apply(lambda x:' '.join(x))
movies_credits['genres'] = movies_credits['genres'].apply(lambda x:' '.join(x
movies_credits['overview'] = movies_credits['overview'].apply(lambda x:' '.jo
movies_credits['production_companies'] = movies_credits['production_companies
movies_credits['production_countries'] = movies_credits['production_countries
movies_credits['keywords'] = movies_credits['keywords'].apply(lambda x:' '.jo
```

In [170]:
```
movies_credits.head(1)
```

Out[170]:

	budget	genres	id	imdb_id	original_language	overview	popularity	
0	30000000	Animation Comedy Family	862	114709	English	Led by Woody, Andy's toys live happily in his ...	21.946943	/rhIRbceoE9lR4veE

1 rows × 24 columns

In [171]:
```
movies_credits['features'] = movies_credits['features'].apply(lambda x:x.lowe
```

In [175]:
```
#Reading the dataset
movies_credits.to_csv(r"G:\rec_sys\movies_credits.csv")
```

In [176]:
```
#Saving the dataset in csv format
ratings2.to_csv(r"G:\rec_sys\ratings2.csv")
```

In [177]:
```
movies_credits.isnull().sum()
```

Out[177]:
```
budget                     0
genres                     0
id                         0
imdb_id                    0
original_language          0
overview                   0
popularity                 0
poster_path              204
production_companies       0
production_countries       0
release_date               0
revenue                    0
runtime                    0
tagline                16193
title                      0
vote_average               0
vote_count                 0
cast                       0
crew                       0
movieId                    0
keywords                   0
release_year               0
imdbid                    12
features                   0
dtype: int64
```

In [206]:
```
df = pd.read_csv(r"G:\rec_sys\movies_credits.csv")
```

In [187]:
```
df
```

Out[187]:

	Unnamed: 0	budget	genres	id	imdb_id	original_language	overview	p
0	0	30000000	Animation Comedy Family	862	114709	English	Led by Woody, Andy's toys live happily in his ...	2
1	1	65000000	Adventure Fantasy Family	8844	113497	English	When siblings Judy and Peter discover an encha...	1
2	2	0	Romance Comedy	15602	113228	English	A family wedding reignites the ancient feud be...	1
3	3	16000000	Comedy Drama Romance	31357	114885	English	Cheated on, mistreated and stepped on, the wom...	
4	4	0	Comedy	11862	113041	English	Just when George Banks has recovered from his ...	
...	...	...	...	...	...	...	...	
30897	31082	0	Horror	289923	252966	English	A film archivist revisits the story of Rustin ...	
30898	31083	0	ScienceFiction	222848	112613	English	It's the year 3000 AD. The world's most danger...	
30899	31084	0	Drama Action Romance	30840	102797	English	Yet another version of the classic epic, with ...	
30900	31085	0	Action Drama Thriller	67758	303758	English	When one of her hits goes wrong, a professiona...	
30901	31086	0	NaN	227506	8536	English	In a small town live two brothers, one a minis...	

30902 rows × 25 columns

In [186]:
```
movies_credits
```

Out[186]:

	budget	genres	id	imdb_id	original_language	overview	popularity	
0	30000000	Animation Comedy Family	862	114709	English	Led by Woody, Andy's toys live happily in his ...	21.946943	
1	65000000	Adventure Fantasy Family	8844	113497	English	When siblings Judy and Peter discover an encha...	17.015539	
2	0	Romance Comedy	15602	113228	English	A family wedding reignites the ancient feud be...	11.712900	
3	16000000	Comedy Drama Romance	31357	114885	English	Cheated on, mistreated and stepped on, the wom...	3.859495	/1
4	0	Comedy	11862	113041	English	Just when George Banks has recovered from his ...	8.387519	
...	...	...	...	...	...	...	...	
31082	0	Horror	289923	252966	English	A film archivist revisits the story of Rustin ...	0.386450	
31083	0	ScienceFiction	222848	112613	English	It's the year 3000 AD. The world's most danger...	0.661558	
31084	0	Drama Action Romance	30840	102797	English	Yet another version of the classic epic, with ...	5.683753	/f
31085	0	Action Drama Thriller	67758	303758	English	When one of her hits goes wrong, a professiona...	0.903007	/c
31086	0		227506	8536	English	In a small town live two brothers, one a minis...	0.003503	

30902 rows × 24 columns

In [244]:
```
df.isnull().sum()
```

Out[244]:
```
Unnamed: 0                 0
budget                     0
genres                  1580
id                         0
imdb_id                    0
original_language          0
overview                 445
popularity                 0
poster_path                0
production_companies       0
production_countries       0
release_date               0
revenue                    0
runtime                    0
tagline                15990
title                      0
vote_average               0
vote_count                 0
cast                    1122
crew                     391
movieId                    0
keywords                8543
release_year               0
imdbid                    12
features                   0
dtype: int64
```

In [210]:
```
df.dropna(subset='features',inplace=True)#drop the nan values
```

In [211]:
```
df['features'].isnull().sum()
```

Out[211]:
```
0
```

In [217]:
```
df[df['production_countries'].isnull()]
```

Out[217]:

	Unnamed: 0	budget	genres	id	imdb_id	original_language	
51	51	0	Action Thriller Drama	117164	109950	English	Detectiv bodygu
56	56	0	NaN	124057	113541	English	Set in mo Alex finds K
84	84	0	NaN	188588	113612	English	Filmec loca Ha
107	108	0	Documentary	89333	112646	English	A do follov Turlir
108	109	0	Crime	96357	113276	English	An ex- grou ho:

In [216]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

In [ ]:
```
#Filling the nan with mode values from 'production_countries' and 'production_
```

In [226]:
```
df['production_countries'].replace(np.NaN,df['production_countries'].mode()[0
```

In [241]:
```
df.dropna(subset='poster_path',inplace=True)
```

In [ ]:
```
df.dropna(subset='poster_path',inplace=True)
```

In [243]:
```
df['production_companies'].replace(np.NaN,df['production_companies'].mode()[0
```

In [245]:
```
df.to_csv(r"G:\rec_sys\movies_credits1.csv")
```

In [246]:
```
df1 = pd.read_csv(r"G:\rec_sys\ratings2.csv")
```

In [247]:
```
df1.isnull().sum()
```

Out[247]:
```
Unnamed: 0      0
title           0
movieId         0
release_year    0
rating          0
dtype: int64
```

In [248]:
```
df1.dtypes
```

Out[248]:
```
Unnamed: 0        int64
title            object
movieId           int64
release_year      int64
rating          float64
dtype: object
```

In [252]:
```
df.dtypes
```

Out[252]:
```
Unnamed: 0                      int64
budget                          int64
genres                         object
id                              int64
imdb_id                         int64
original_language              object
overview                       object
popularity                    float64
poster_path                    object
production_companies           object
production_countries           object
release_date           datetime64[ns]
revenue                       float64
runtime                       float64
tagline                        object
title                          object
vote_average                  float64
vote_count                    float64
cast                           object
crew                           object
movieId                         int64
keywords                       object
release_year                    int64
imdbid                         object
features                       object
dtype: object
```

In [251]:
```
df['release_date'] = pd.to_datetime(df['release_date'])
```

Saved the cleaned dataset in csv format

In [255]:
```
df.to_csv(r"G:\rec_sys\movies.csv")
```

In [ ]:
```

```