Importing Libraries

In [206…
```
import pandas as pd
import requests
import warnings
warnings.simplefilter('ignore')
```

In [207…
```
#creating an empty Dataframe
df = pd.DataFrame()
```

Out[207]: —

Fetching movies data from TMDB using its API

In [208…
```
# using for loop to fetch the movie data
# range provided indicates the page number given by the TMDB, each page has 20 movies data
# created a dataframe into for loop and then concatenate the fetched data into dataframe called 'df'
for i in range(1,501):
    url = "https://api.themoviedb.org/3/discover/movie?api_key=7bca32596da7d3ef9aa511c95aee829b&language=en-US&region=IN&page={}".format(i)
    headers = {    "accept": "application/json",
    "Authorization": "Bearer eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI3YmNhMzI1OTZkYTdkM2VmOWFhNTExYzk1YWVlODI5YiIsInN1YiI6IjY0YzBmMWYwMWNmZTNhMGViMzBiZGU3MyIsInNjb3BlcyI6WyJhcGlfcmVhZCJdLCJ2ZXJzaW9uIjoxfQ.az1FVt8lcPgNY4BCPOFqBso-bA4liYODgeQzf
    response = requests.get(url, headers=headers)
    temp_df = pd.DataFrame(response.json()['results'])[['adult', 'genre_ids', 'id', 'original_language', 'title', 'overview']]
    frames = (df , temp_df)
    df = pd.concat(frames, ignore_index = True)
```

In [209…
```
#creating a copy
movie = df.copy()
```

In [210…
```
movie
```

Out[210]:

	adult	genre_ids	id	original_language	title	overview
0	False	[28, 12, 878]	298618	en	The Flash	When his attempt to save his family inadverten...
1	False	[35, 12, 14]	346698	en	Barbie	Barbie and Ken are having the time of their li...
2	False	[28, 12, 878]	667538	en	Transformers: Rise of the Beasts	When a new threat capable of destroying the en...
3	False	[28, 80, 53]	385687	en	Fast X	Over many missions and against impossible odds...
4	False	[18, 36]	872585	en	Oppenheimer	The story of J. Robert Oppenheimer's role in t...
...	...	...	...	...	...	...
9995	False	[10749, 35]	506180	ta	Yettikki Potti	Comedy Tale
9996	False	[28, 18]	262489	ml	Salaam Kashmier	Salaam Kashmir revolves around two men - Tomy ...
9997	False	[10749, 28, 18]	578357	bn	Deewana	Abhi (Jeet) is a funloving and extremely popul...
9998	False	[80]	543954	hi	Hanste Khelte	Three friends who don't take their life seriou...
9999	False	[35, 18, 10751]	339978	ta	Parijatham	Prakashraj and Seetha have a son named Surendh...

10000 rows × 6 columns

In [211…
```
# using for loop to fetch the genre data for movies
genre = requests.get('https://api.themoviedb.org/3/genre/movie/list?api_key=7bca32596da7d3ef9aa511c95aee829b&language=en-US')
genre = genre.json()['genres']
genre
```

Out[211]:
```
[{'id': 28, 'name': 'Action'},
 {'id': 12, 'name': 'Adventure'},
 {'id': 16, 'name': 'Animation'},
 {'id': 35, 'name': 'Comedy'},
 {'id': 80, 'name': 'Crime'},
 {'id': 99, 'name': 'Documentary'},
 {'id': 18, 'name': 'Drama'},
 {'id': 10751, 'name': 'Family'},
 {'id': 14, 'name': 'Fantasy'},
 {'id': 36, 'name': 'History'},
 {'id': 27, 'name': 'Horror'},
 {'id': 10402, 'name': 'Music'},
 {'id': 9648, 'name': 'Mystery'},
 {'id': 10749, 'name': 'Romance'},
 {'id': 878, 'name': 'Science Fiction'},
 {'id': 10770, 'name': 'TV Movie'},
 {'id': 53, 'name': 'Thriller'},
 {'id': 10752, 'name': 'War'},
 {'id': 37, 'name': 'Western'}]
```

In [ ]:
```
# creating a list that contains the genre of the movie in an arranged manner as per the movies stored in our dataframe
genre_list = []
for i in movie['genre_ids']:
    l = []
    for j in i:
        for k in genre:
            if k['id'] == j:
                l.append(k['name'])
    genre_list.append(l)
```

In [213…
```
movie['genre'] = genre_list
```

In [214…
```
movie
```

Out[214]:

	adult	genre_ids	id	original_language	title	overview	genre
0	False	[28, 12, 878]	298618	en	The Flash	When his attempt to save his family inadverten...	[Action, Adventure, Science Fiction]
1	False	[35, 12, 14]	346698	en	Barbie	Barbie and Ken are having the time of their li...	[Comedy, Adventure, Fantasy]
2	False	[28, 12, 878]	667538	en	Transformers: Rise of the Beasts	When a new threat capable of destroying the en...	[Action, Adventure, Science Fiction]
3	False	[28, 80, 53]	385687	en	Fast X	Over many missions and against impossible odds...	[Action, Crime, Thriller]
4	False	[18, 36]	872585	en	Oppenheimer	The story of J. Robert Oppenheimer's role in t...	[Drama, History]
...	...	...	...	...	...	...	...
9995	False	[10749, 35]	506180	ta	Yettikki Potti	Comedy Tale	[Romance, Comedy]
9996	False	[28, 18]	262489	ml	Salaam Kashmier	Salaam Kashmir revolves around two men - Tomy ...	[Action, Drama]
9997	False	[10749, 28, 18]	578357	bn	Deewana	Abhi (Jeet) is a funloving and extremely popul...	[Romance, Action, Drama]
9998	False	[80]	543954	hi	Hanste Khelte	Three friends who don't take their life seriou...	[Crime]
9999	False	[35, 18, 10751]	339978	ta	Parijatham	Prakashraj and Seetha have a son named Surendh...	[Comedy, Drama, Family]

10000 rows × 7 columns

In [215…
```
# dropping the unwanted columns
movie.drop(columns = ['adult', 'genre_ids'],inplace = True)
```

In [216…
```
# using for loop to fetch the TV series data
# range provided indicates the page number given by the TMDB, each page has 20 TV series data
# created a dataframe into for loop and then concatenate the fetched data into dataframe called 'df'
TV = pd.DataFrame()
for i in range(1,501):
    url = "https://api.themoviedb.org/3/discover/tv?include_adult=true&page={}&watch_region=IN".format(i)
    headers = {    "accept": "application/json",
    "Authorization": "Bearer eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI3YmNhMzI1OTZkYTdkM2VmOWFhNTExYzk1YWVlODI5YiIsInN1YiI6IjY0YzBmMWYwMWNmZTNhMGViMzBiZGU3MyIsInNjb3BlcyI6WyJhcGlfcmVhZCJdLCJ2ZXJzaW9uIjoxfQ.az1FVt8lcPgNY4BCPOFqBso-bA4liYODgeQzf
    response = requests.get(url, headers=headers)
    temp_df1 = pd.DataFrame(response.json()['results'])[['id','original_language','name','overview','genre_ids']]
    frames = (TV, temp_df1)
    TV = pd.concat(frames, ignore_index = True)
```

In [217…
```
#creating a copy
TV1 = TV.copy()
```

In [218…
```
# using for loop to fetch the genre data for movies
genre1 = requests.get('https://api.themoviedb.org/3/genre/tv/list?api_key=7bca32596da7d3ef9aa511c95aee829b')
genre1 = genre1.json()['genres']
genre1
```

Out[218]:
```
[{'id': 10759, 'name': 'Action & Adventure'},
 {'id': 16, 'name': 'Animation'},
 {'id': 35, 'name': 'Comedy'},
 {'id': 80, 'name': 'Crime'},
 {'id': 99, 'name': 'Documentary'},
 {'id': 18, 'name': 'Drama'},
 {'id': 10751, 'name': 'Family'},
 {'id': 10762, 'name': 'Kids'},
 {'id': 9648, 'name': 'Mystery'},
 {'id': 10763, 'name': 'News'},
 {'id': 10764, 'name': 'Reality'},
 {'id': 10765, 'name': 'Sci-Fi & Fantasy'},
 {'id': 10766, 'name': 'Soap'},
 {'id': 10767, 'name': 'Talk'},
 {'id': 10768, 'name': 'War & Politics'},
 {'id': 37, 'name': 'Western'}]
```

In [219…
```
# creating a list that contains the genre of the movie in an arranged manner as per the movies stored in our dataframe
genre_list_TV = []
for i in TV1['genre_ids']:
    l = []
    for j in i:
        for k in genre1:
            if k['id'] == j:
                l.append(k['name'])
    genre_list_TV.append(l)
```

In [220…
```
# renaming column as well as appending genre list to the dataframe
TV1['genre'] = genre_list_TV
TV1.drop(columns = ['genre_ids'], inplace = True)
TV1.rename(columns = {'name': 'title'}, inplace = True)
TV1
```

Out[220]:

	id	original_language	title	overview	genre
0	94722	de	Tagesschau	German daily news program, the oldest still ex...	[News]
1	209265	pt	Terra e Paixão		[Drama, Crime, Soap]
2	114472	en	Secret Invasion	Nick Fury and Talos discover a faction of shap...	[Drama, Sci-Fi & Fantasy, Action & Adventure]
3	215803	tl	Batang Quiapo	A young man rises to be one of the biggest out...	[Action & Adventure, Comedy, Drama]
4	215546	tl	Luv Is	A Philippine seasonal anthology TV adaptations...	[Drama]
...	...	...	...	...	...
9995	610	en	Mr. Show with Bob and David	Mr. Show with Bob and David is an American ske...	[Comedy]
9996	103578	zh	The Twin Flower Legend	Hua Mujin grew up during the chaotic era of th...	[Drama]
9997	72092	zh	Xuan-Yuan Sword: Han Cloud	After the ancient Emperor defeated the demons,...	[Drama]
9998	23600	en	This Is Love		[]
9999	27119	en	The Comic Strip Presents	The Comic Strip is a group of British comedian...	[Comedy]

10000 rows × 5 columns

In [221…
```
# joining two dataset movie and TV and stored in a  dataframe named 'df'
frames = [movie , TV1]
df = pd.concat(frames, ignore_index = True)
```

In [222…
```
df
```

Out[222]:

	id	original_language	title	overview	genre
0	298618	en	The Flash	When his attempt to save his family inadverten...	[Action, Adventure, Science Fiction]
1	346698	en	Barbie	Barbie and Ken are having the time of their li...	[Comedy, Adventure, Fantasy]
2	667538	en	Transformers: Rise of the Beasts	When a new threat capable of destroying the en...	[Action, Adventure, Science Fiction]
3	385687	en	Fast X	Over many missions and against impossible odds...	[Action, Crime, Thriller]
4	872585	en	Oppenheimer	The story of J. Robert Oppenheimer's role in t...	[Drama, History]
...	...	...	...	...	...
19995	610	en	Mr. Show with Bob and David	Mr. Show with Bob and David is an American ske...	[Comedy]
19996	103578	zh	The Twin Flower Legend	Hua Mujin grew up during the chaotic era of th...	[Drama]
19997	72092	zh	Xuan-Yuan Sword: Han Cloud	After the ancient Emperor defeated the demons,...	[Drama]
19998	23600	en	This Is Love		[]
19999	27119	en	The Comic Strip Presents	The Comic Strip is a group of British comedian...	[Comedy]

20000 rows × 5 columns

Fetching Keywords for movies as well as TV series from TMDB

In [223…
```
# creating a list which has id values of all movies
movie_lst = []
for i in (movie['id']):
    movie_lst.append(i)
```

In [258…
```
# creating an empty list
keys1 = []
```

Using for loop to fetch the data from TMDB API. It has been done in parts to understand the progress as it takes a long time to load. Indexing is performed to fetch the movies id in arranged manner

In [259…
```
for i in (movie_lst[0:2001]):
    key = requests.get("https://api.themoviedb.org/3/movie/{}/keywords?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    key = key.json()['keywords']
    keys1.append(key)
```

In [260…
```
for i in (movie_lst[2001:4001]):
    key = requests.get("https://api.themoviedb.org/3/movie/{}/keywords?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    key = key.json()['keywords']
    keys1.append(key)
```

In [261…
```
for i in (movie_lst[4001:6001]):
    key = requests.get("https://api.themoviedb.org/3/movie/{}/keywords?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    key = key.json()['keywords']
    keys1.append(key)
```

In [262…
```
for i in (movie_lst[6001:8001]):
    key = requests.get("https://api.themoviedb.org/3/movie/{}/keywords?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    key = key.json()['keywords']
    keys1.append(key)
```

In [263…
```
for i in (movie_lst[8001:10001]):
    key = requests.get("https://api.themoviedb.org/3/movie/{}/keywords?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    key = key.json()['keywords']
    keys1.append(key)
```

In [264…
```
movie_keys = keys1.copy()
```

In [230…
```
# creating a list which has id values of all TV series
tv_lst = []
for i in (TV1['id']):
    tv_lst.append(i)
```

In [231…
```
keys2 = []
```

In [232…
```
for i in (tv_lst[0:2001]):
    key = requests.get("https://api.themoviedb.org/3/tv/{}/keywords?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    key = key.json()['results']
    keys2.append(key)
```

In [233…
```
for i in (tv_lst[2001:4001]):
    key = requests.get("https://api.themoviedb.org/3/tv/{}/keywords?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    key = key.json()['results']
    keys2.append(key)
```

In [234…
```
for i in (tv_lst[4001:6001]):
    key = requests.get("https://api.themoviedb.org/3/tv/{}/keywords?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    key = key.json()['results']
    keys2.append(key)
```

In [235…
```
for i in (tv_lst[6001:8001]):
    key = requests.get("https://api.themoviedb.org/3/tv/{}/keywords?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    key = key.json()['results']
    keys2.append(key)
```

In [236…
```
for i in (tv_lst[8001:10001]):
    key = requests.get("https://api.themoviedb.org/3/tv/{}/keywords?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    key = key.json()['results']
    keys2.append(key)
```

In [265…
```
tv_keys = keys2.copy()
```

In [266…
```
# appending one list into another
movie_keys.extend(tv_keys)
```

In [268…
```
final_keys = movie_keys.copy()
len(final_keys)
```

Out[268]: 20000

In [269…
```
keywords = []
for i in final_keys:
    list_123 = []
    for j in i:
        list_123.append(j['name'])
    keywords.append(list_123)
```

In [271…
```
final_keywords = keywords.copy()
```

In [272…
```
# creating a column named keyword in df and appending keywords list into the dataframe
df['keywords'] = final_keywords
```

Fetching Cast & Crew data from TMDB API

In [297…
```
# creating an empty lists.
cast_lst = []
crew_lst = []
```

Using for loop to fetch the data from TMDB API. It has been done in parts to understand the progress as it takes a long time to load. Indexing is performed to fetch the movies id in arranged manner

In [298…
```
for i in (movie_lst[0:2001]):
    key = requests.get("https://api.themoviedb.org/3/movie/{}/credits?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    cast_key = key.json()['cast']
    crew_key = key.json()['crew']
    cast_lst.append(cast_key)
    crew_lst.append(crew_key)
```

In [299…
```
for i in (movie_lst[2001:4001]):
    key = requests.get("https://api.themoviedb.org/3/movie/{}/credits?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    cast_key = key.json()['cast']
    crew_key = key.json()['crew']
    cast_lst.append(cast_key)
    crew_lst.append(crew_key)
```

In [300…
```
for i in (movie_lst[4001:6001]):
    key = requests.get("https://api.themoviedb.org/3/movie/{}/credits?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    cast_key = key.json()['cast']
    crew_key = key.json()['crew']
    cast_lst.append(cast_key)
    crew_lst.append(crew_key)
```

In [301…
```
for i in (movie_lst[6001:8001]):
    key = requests.get("https://api.themoviedb.org/3/movie/{}/credits?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    cast_key = key.json()['cast']
    crew_key = key.json()['crew']
    cast_lst.append(cast_key)
    crew_lst.append(crew_key)
```

In [302…
```
for i in (movie_lst[8001:10001]):
    key = requests.get("https://api.themoviedb.org/3/movie/{}/credits?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    cast_key = key.json()['cast']
    crew_key = key.json()['crew']
    cast_lst.append(cast_key)
    crew_lst.append(crew_key)
```

In [343…
```
cast_lst1 = cast_lst.copy()
crew_lst1 = crew_lst.copy()
```

Same thing is repeated for TV series

In [328…
```
cast_lst2 = []
crew_lst2 = []
```

In [329…
```
for i in (tv_lst[0:2001]):
    key = requests.get("https://api.themoviedb.org/3/tv/{}/credits?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    cast_key = key.json()['cast']
    crew_key = key.json()['crew']
    cast_lst2.append(cast_key)
    crew_lst2.append(crew_key)
```

In [330…
```
for i in (tv_lst[2001:4001]):
    key = requests.get("https://api.themoviedb.org/3/tv/{}/credits?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    cast_key = key.json()['cast']
    crew_key = key.json()['crew']
    cast_lst2.append(cast_key)
    crew_lst2.append(crew_key)
```

In [331…
```
for i in (tv_lst[4001:6001]):
    key = requests.get("https://api.themoviedb.org/3/tv/{}/credits?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    cast_key = key.json()['cast']
    crew_key = key.json()['crew']
    cast_lst2.append(cast_key)
    crew_lst2.append(crew_key)
```

In [332…
```
for i in (tv_lst[6001:8001]):
    key = requests.get("https://api.themoviedb.org/3/tv/{}/credits?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    cast_key = key.json()['cast']
    crew_key = key.json()['crew']
    cast_lst2.append(cast_key)
    crew_lst2.append(crew_key)
```

In [333…
```
for i in (tv_lst[8001:10001]):
    key = requests.get("https://api.themoviedb.org/3/tv/{}/credits?api_key=7bca32596da7d3ef9aa511c95aee829b".format(i))
    cast_key = key.json()['cast']
    crew_key = key.json()['crew']
    cast_lst2.append(cast_key)
    crew_lst2.append(crew_key)
```

In [340…
```
cast_lst_tv = cast_lst2.copy()
crew_lst_tv = crew_lst2.copy()
```

In [346…
```
cast_lst1.extend(cast_lst_tv)
```

In [347…
```
crew_lst1.extend(crew_lst_tv)
```

In [350…
```
df['cast'] = cast_lst1
df['crew'] = crew_lst1
```

We have cast data as shown below but the desired data we need is in lists

To create a list we will define a function, that will create a list which contains name of actors from the available json format.

In [353…
```
def fetch_actors(obj):
    L = []
    for i in obj:
        if i['known_for_department'] == 'Acting':
            L.append(i['name'])
    return L
```

In [355…
```
# using apply function to apply our defined function
df['cast'] = df['cast'].apply(fetch_actors)
```

To create a list we will define a function, that will create a list which contains name of director from the available json format.

In [360…
```
def fetch_director(obj):
    L = []
    for i in obj:
        if i['job'] == 'Director':
            L.append(i['name'])
    return L
```

In [361…
```
# using apply function to apply our defined function
df['crew'] = df['crew'].apply(fetch_director)
```

In [362…
```
df
```

Out[362]:

	id	original_language	title	overview	genre	keywords	cast	crew
0	298618	en	The Flash	When his attempt to save his family inadverten...	[Action, Adventure, Science Fiction]	[secret identity, hero, superhero, based on co...	[Ezra Miller, Sasha Calle, Michael Keaton, Mic...	[Andy Muschietti]
1	346698	en	Barbie	Barbie and Ken are having the time of their li...	[Comedy, Adventure, Fantasy]	[feminism, patriarchy, satire, based on toy, f...	[Margot Robbie, Ryan Gosling, Issa Rae, Kate M...	[Greta Gerwig]
2	667538	en	Transformers: Rise of the Beasts	When a new threat capable of destroying the en...	[Action, Adventure, Science Fiction]	[peru, alien, end of the world, based on toy, ...	[Anthony Ramos, Dominique Fishback, Peter Cull...	[Steven Caple Jr.]
3	385687	en	Fast X	Over many missions and against impossible odds...	[Action, Crime, Thriller]	[sequel, revenge, racing, family, cars]	[Vin Diesel, Michelle Rodriguez, Tyrese Gibson...	[Louis Leterrier]
4	872585	en	Oppenheimer	The story of J. Robert Oppenheimer's role in t...	[Drama, History]	[based on novel or book, husband wife relation...	[Cillian Murphy, Emily Blunt, Matt Damon, Robe...	[Christopher Nolan]
...	...	...	...	...	...	...	...	...
19995	610	en	Mr. Show with Bob and David	Mr. Show with Bob and David is an American ske...	[Comedy]	[drugs, sketch comedy, dorky]	[Bob Odenkirk, David Cross, Jill Talley, Brian...	[]
19996	103578	zh	The Twin Flower Legend	Hua Mujin grew up during the chaotic era of th...	[Drama]	[friendship, based on novel or book, romance, ...	[Kai Xuan, Mao Xiaohui, Jim Yu, Kenny Kwan, Ch...	[Wu Jin Yuan]
19997	72092	zh	Xuan-Yuan Sword: Han Cloud	After the ancient Emperor defeated the demons,...	[Drama]	[china, romance, based on video game, xianxia]	[Leon Zhang, Alan Yu Menglong, Guan Xiaotong, ...	[반문걸, Sha Jin, Poon Man-kit]
19998	23600	en	This Is Love		[]	[]	[Choi Kang-hee, Kim Ja-ok, Lee Yu-ri]	[]
19999	27119	en	The Comic Strip Presents	The Comic Strip is a group of British comedian...	[Comedy]	[surreal, parody, anthology]	[Rik Mayall, Dawn French, Adrian Edmondson, Je...	[]

20000 rows × 8 columns

In [368…
```
df.to_csv('TMDB_Movies&TV.csv')
```

In [ ]: