# How does a Recommendation System Work?

Recommendation systems use specialized algorithms and [machine learning solutions](https://marutitech.com/machine-learning-services/?utm_source=medium&utm_medium=content_promotion&utm_campaign=Types_Of_Recommendation_Systems). Driven by the automated configuration, coordination, and management of [machine learning predictive analytics algorithms](https://marutitech.com/machine-learning-predictive-analytics/?utm_source=medium&utm_medium=content_promotion&utm_campaign=Types_Of_Recommendation_Systems), the recommendation system can wisely select which filters to apply to a particular user's specific situation. It facilitates marketers to maximize conversions and average order value.

Recommender systems can forecast user ratings, even before they have provided one, making them an effective tool. Mainly, a recommendation system processes data through four phases as follows-

* **Collection**

Data collected can be explicit (ratings and comments on products) or implicit (page views, order history, etc.).

* **Storing**

The type of data used to create recommendations can help you decide the kind of storage you should use- NoSQL database, object storage, or standard SQL database.

* **Analyzing**

The recommender system finds items with similar user engagement data after analysis.

* **Filtering**

This is the last step where data gets filtered to access the relevant information required to provide recommendations to the user. To enable this, you will need to choose an algorithm suiting the recommendation system.

**What are recommender systems?**

Recommender systems are trained to understand the preferences, previous decisions, and characteristics of people and products using data gathered about their interactions. These include impressions, clicks, likes, and purchases. Because of their capability to predict consumer interests and desires on a highly personalized level, recommender systems are a favourite with content and product providers.

**Types Of Recommendation Systems**

[Machine learning solves many problems](https://marutitech.com/problems-solved-machine-learning/?utm_source=medium&utm_medium=content_promotion&utm_campaign=Types_Of_Recommendation_Systems) but making product recommendations is a widely known application of machine learning. There are three main types of recommendation systems –

**1. Collaborative Filtering**

The collaborative filtering method is based on gathering and analyzing data on user’s behavior. This includes the user’s online activities and predicting what they will like based on the similarity with other users.



For example, if user A likes Apple, Banana, and Mango while user B likes Apple, Banana, and Jackfruit, they have similar interests. So, it is highly likely that A would like Jackfruit and B would enjoy Mango. This is how collaborative filtering takes place.

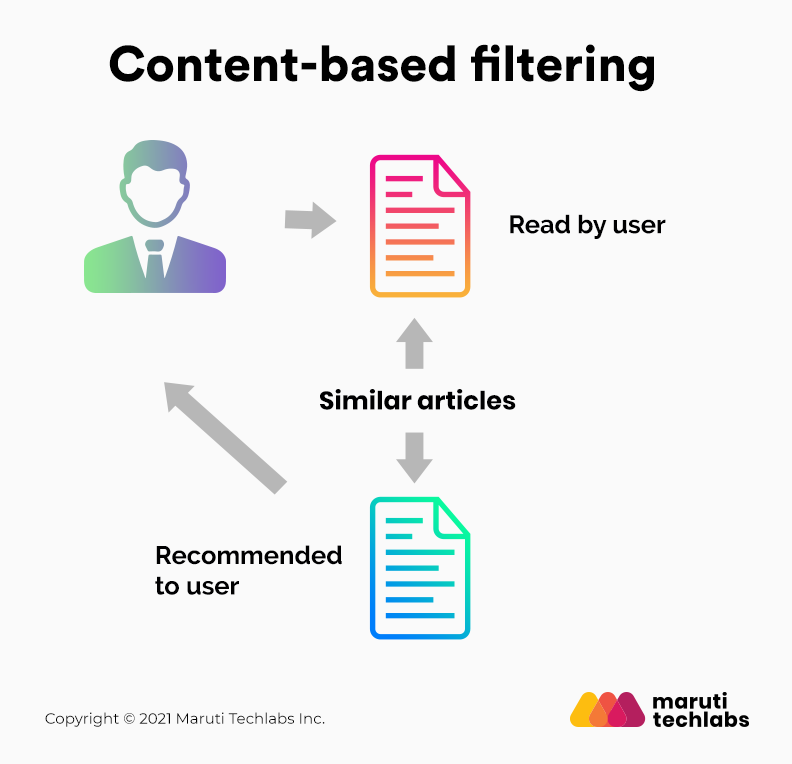
*Two kinds of collaborative filtering techniques used are:*

* User-User collaborative filtering
* Item-Item collaborative filtering

One of the main advantages of this recommendation system is that it can recommend complex items precisely without understanding the object itself. There is no reliance on machine analyzable content.

**2. Content-Based Filtering**

Content-based filtering methods are based on the description of a product and a profile of the user’s preferred choices. In this recommendation system, products are described using keywords, and a user profile is built to express the kind of item this user likes.

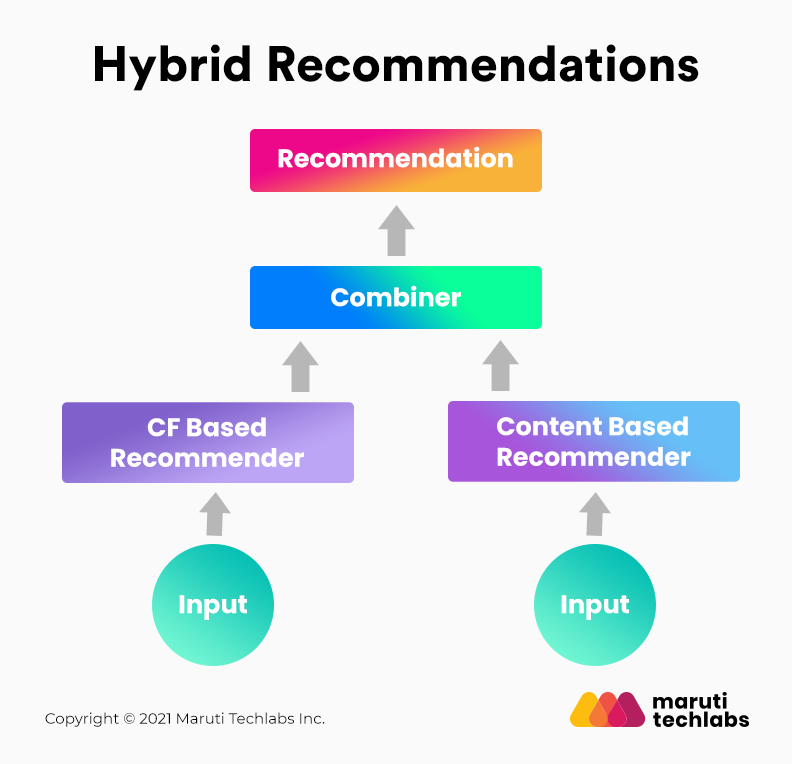


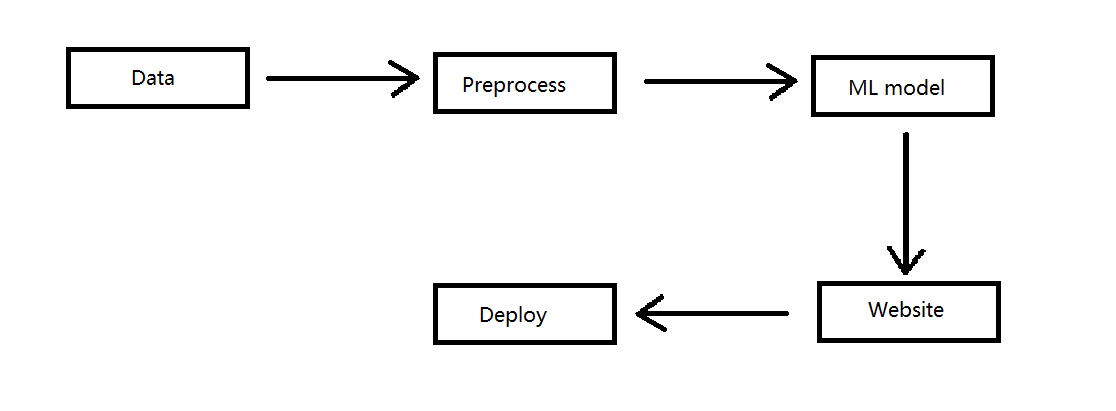
For instance, if a user likes to watch movies such as Iron Man, the recommender system recommends movies of the superhero genre or films describing Tony Stark.

The central assumption of content-based filtering is that you will also like a similar item if you like a particular item.

**3. Hybrid Recommendation Systems**

In hybrid recommendation systems, products are recommended using both content-based and collaborative filtering simultaneously to suggest a broader range of products to customers. This recommendation system is up-and-coming and is said to provide more accurate recommendations than other recommender systems.



Project strategy:  


Technologies used:

* 1. Streamlit: Streamlit is an open-source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.
  2. What is use of pickle in Python?

Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python can be pickled so that it can be saved on disk.

Libraries used:

* streamlit: A Python library for creating web applications with minimal code.
* pickle: Used for serializing and deserializing Python objects (saving and loading).
* pandas: A data manipulation library used for handling data in tabular form.
* requests: Used for making HTTP requests.

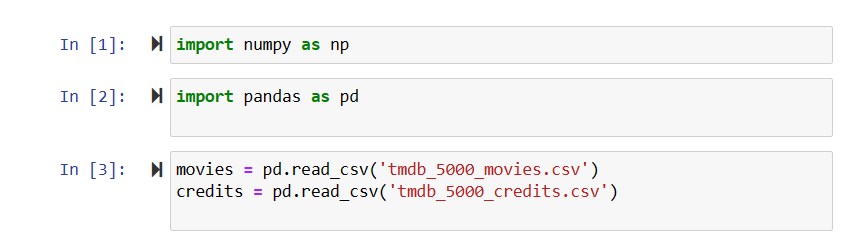
Dataset chosen:

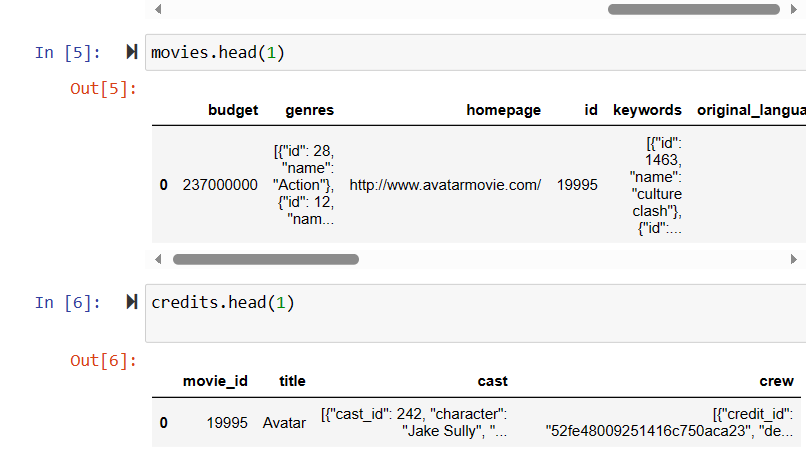
<https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata/data?select=tmdb_5000_movies.csv>

It has data for 5000 movies.

The above data has 2 datasets in it.

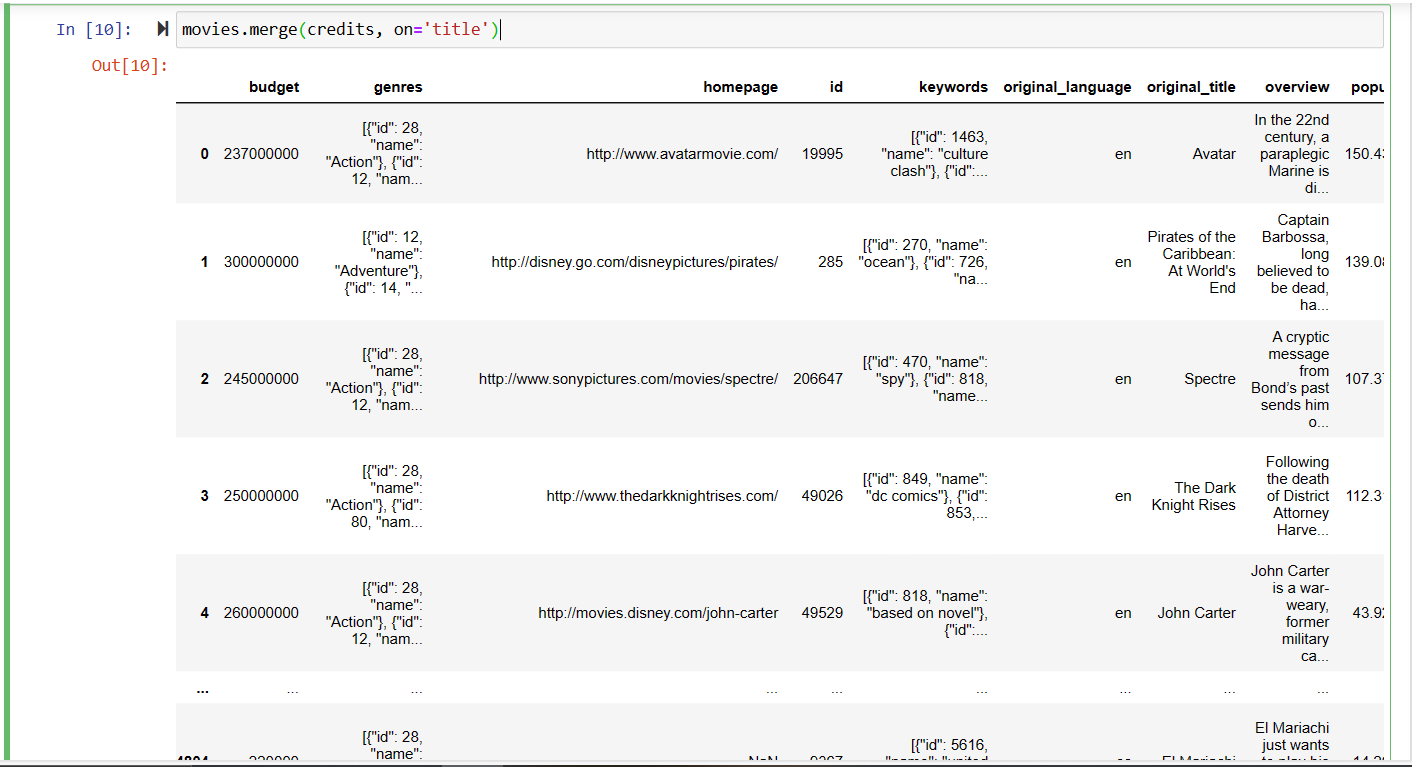
* 1. tmdb\_5000\_credits.csv
  2. tmdb\_5000\_movies.csv

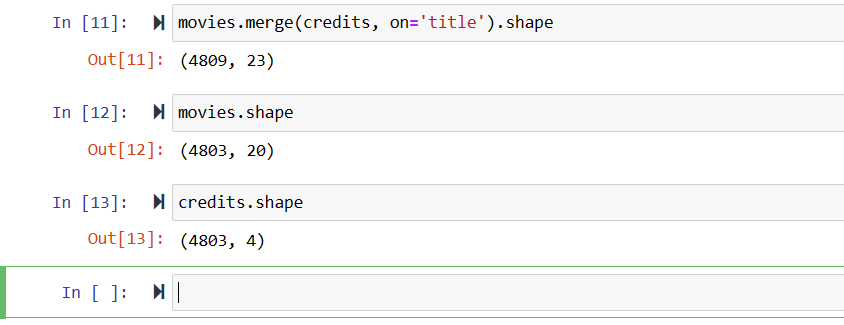


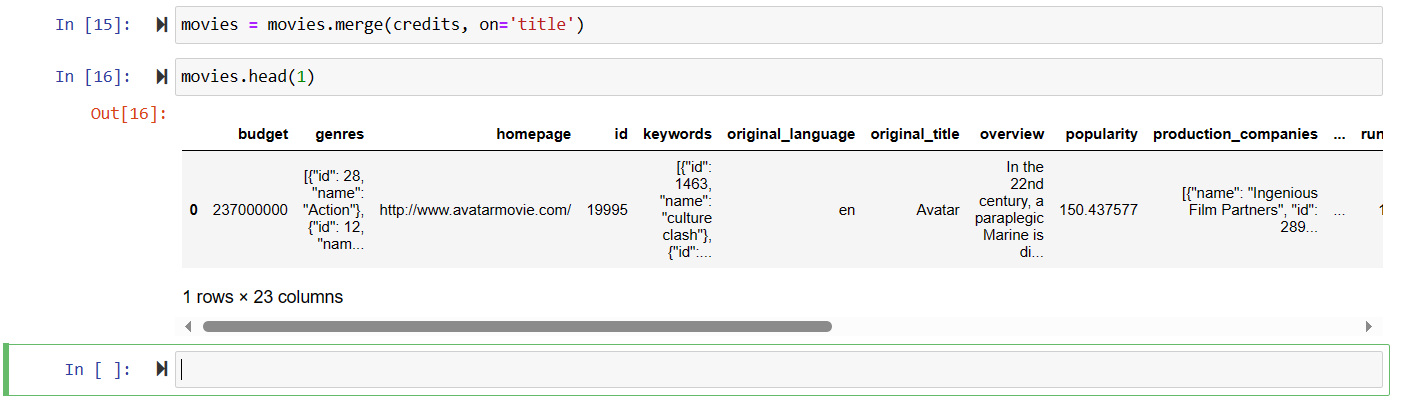


We’ll combine both these datasets, they can be combined on basis of movie id, title.

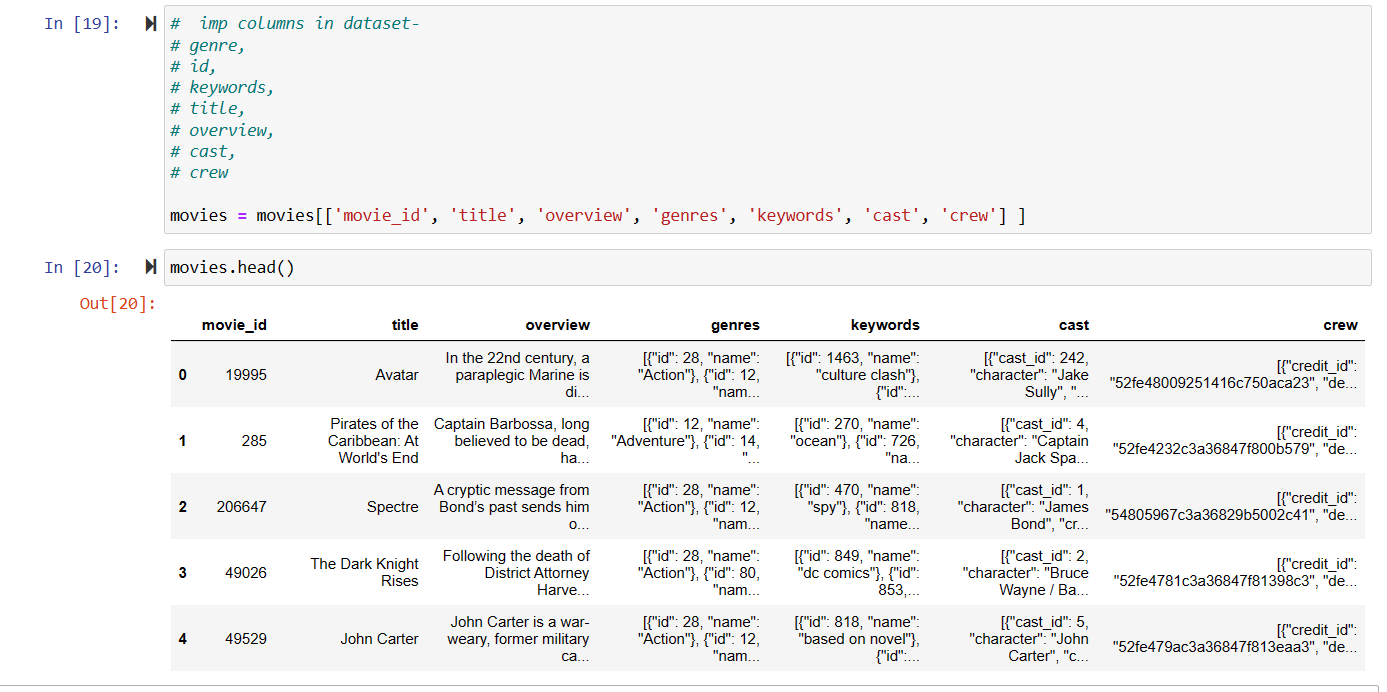
Let’s combine on basis of movie title. Saving the new data in movies itself.







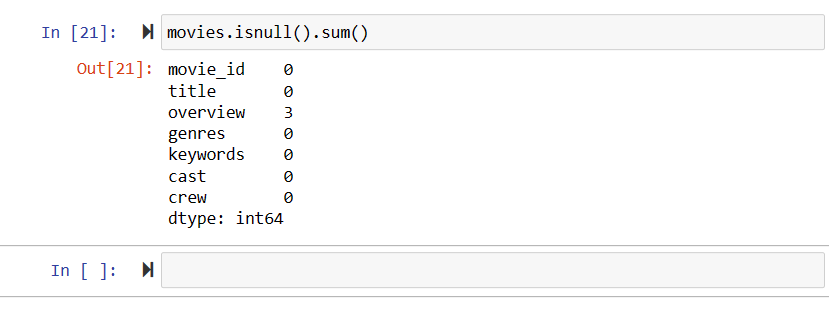
Selecting only the required columns:



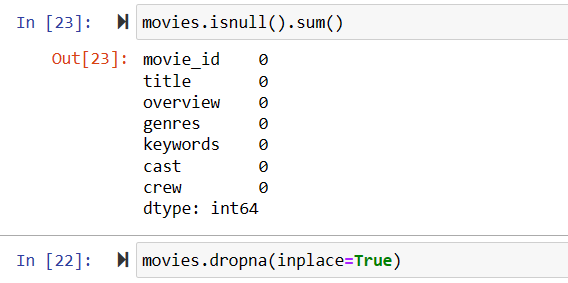
Now we have to create a new dataframe which has these columns:  
Movie\_Id, Movie\_title, Tags.

To make ‘Tags’ column we’ll merge – overview, genres, keywords, cast (only top 3 actors), crew (only director name).

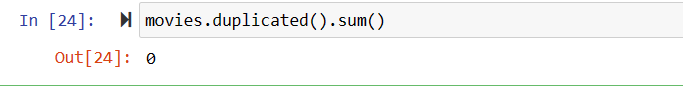
To do this we have to preprocess all, remove duplicate, empty data.



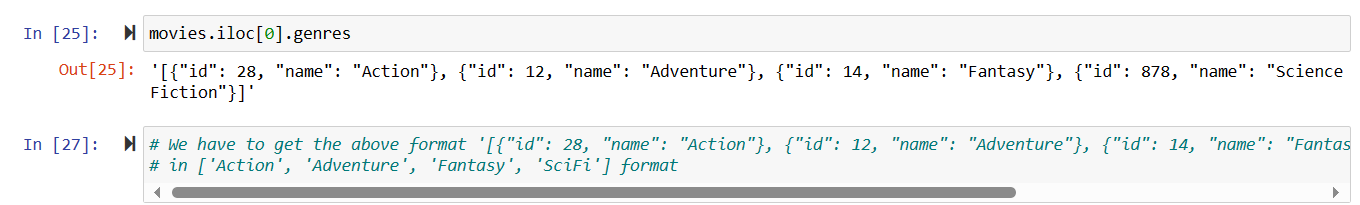
Removing all the null data

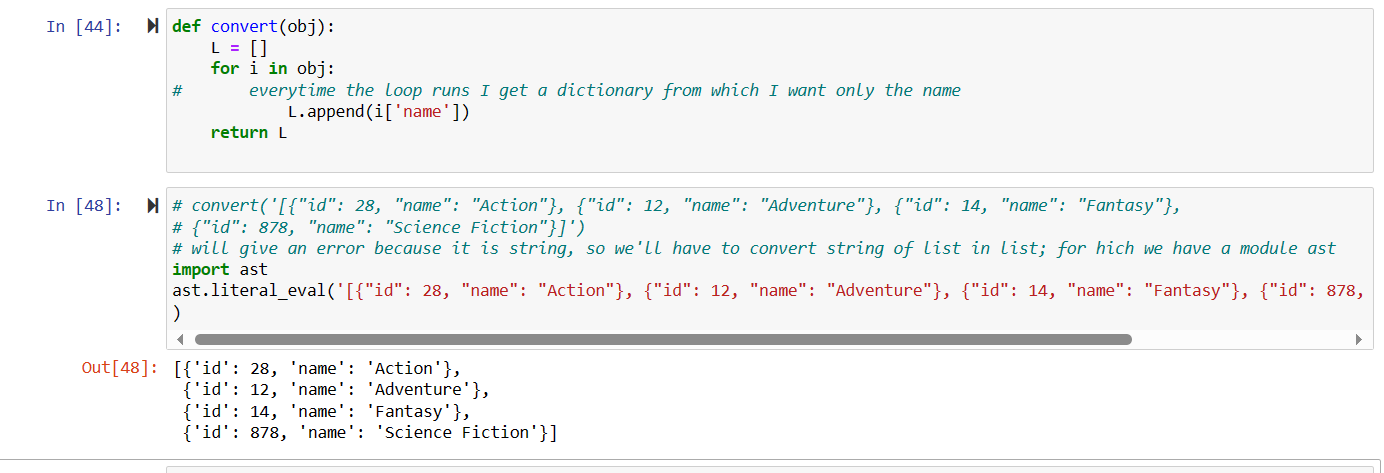


To check if any duplicate data exists:

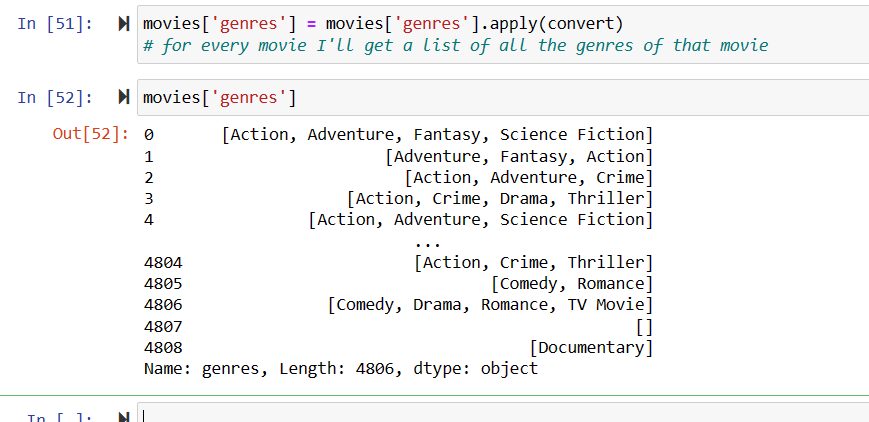


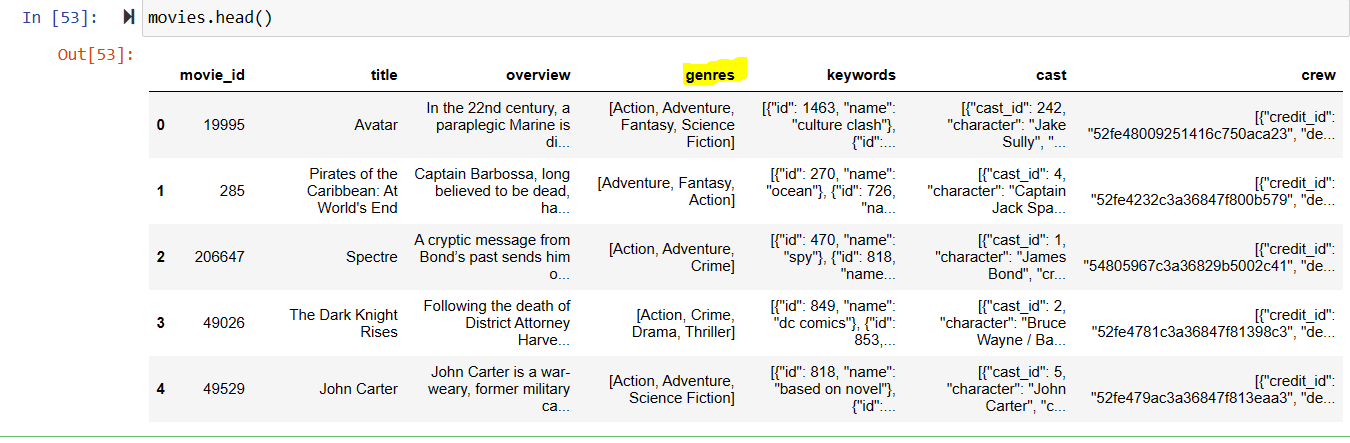
Genres of ‘Avatar’ movie



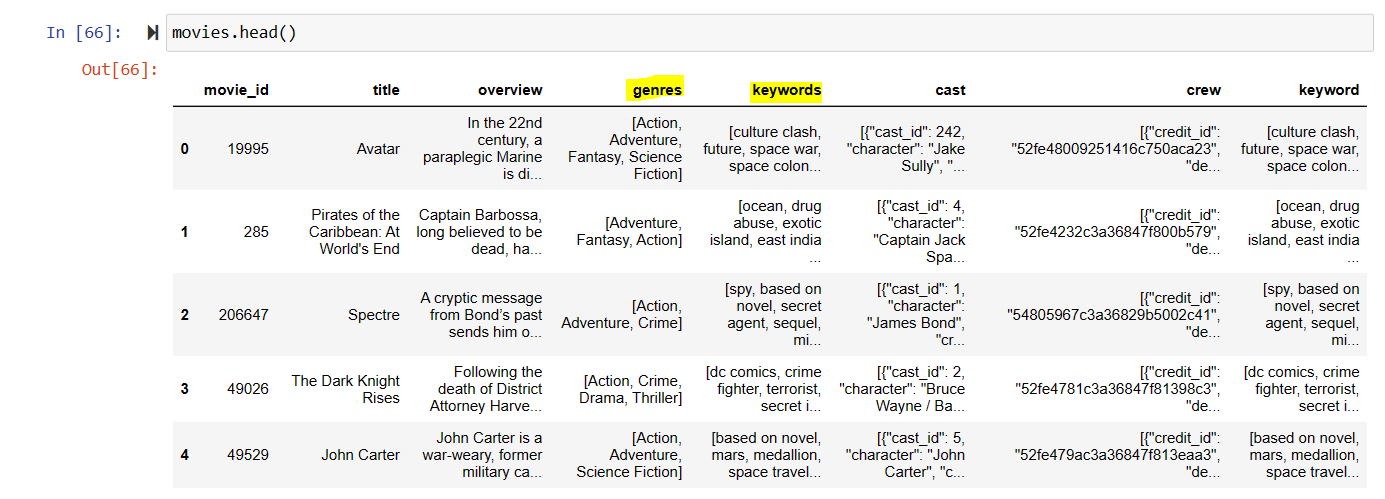


# For every movie I'll get a list of all the genres of that movie

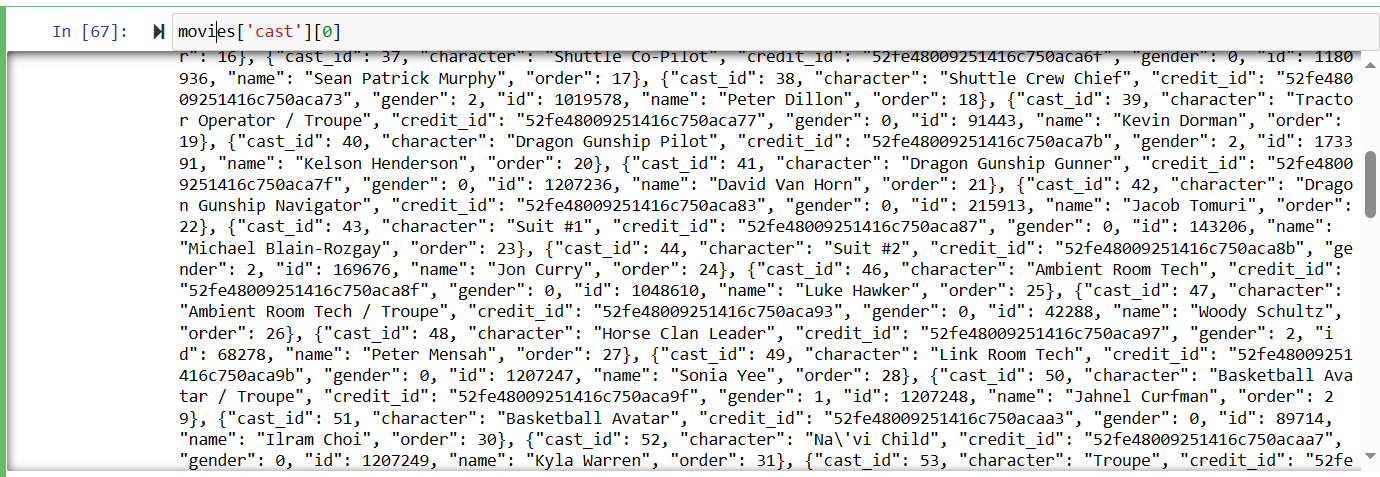


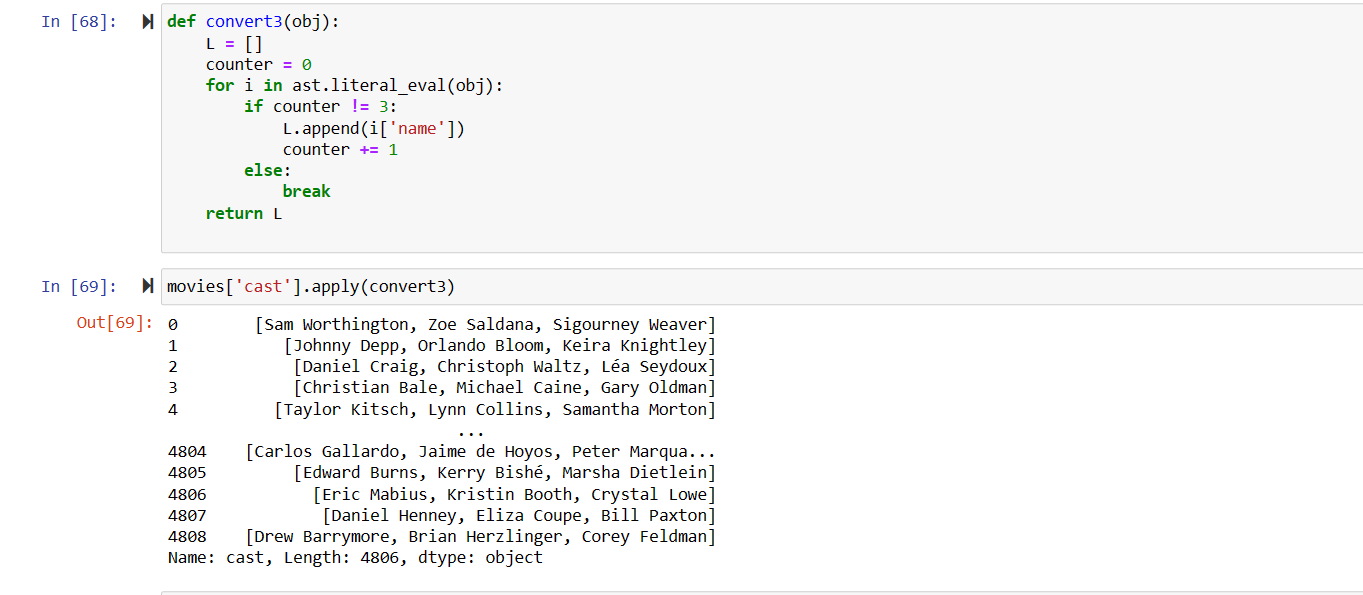
In the genres column we get list of all the respective movies (do same for other columns):  


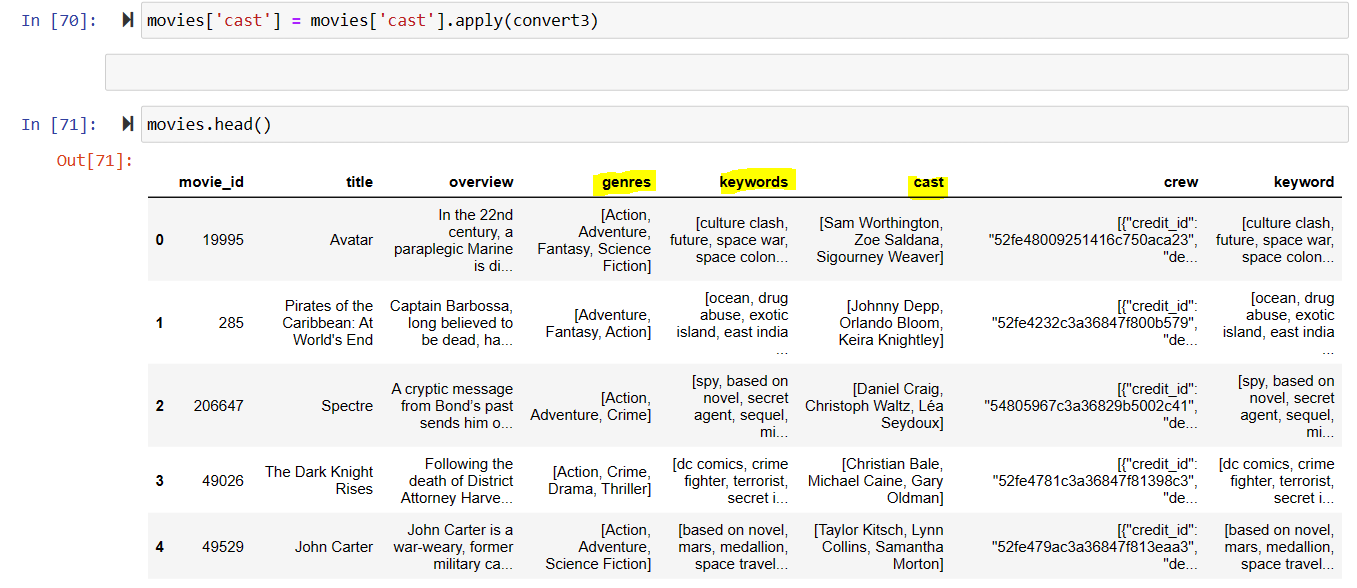
# movies[‘keywords’] = movies[‘keywords’].apply(convert)



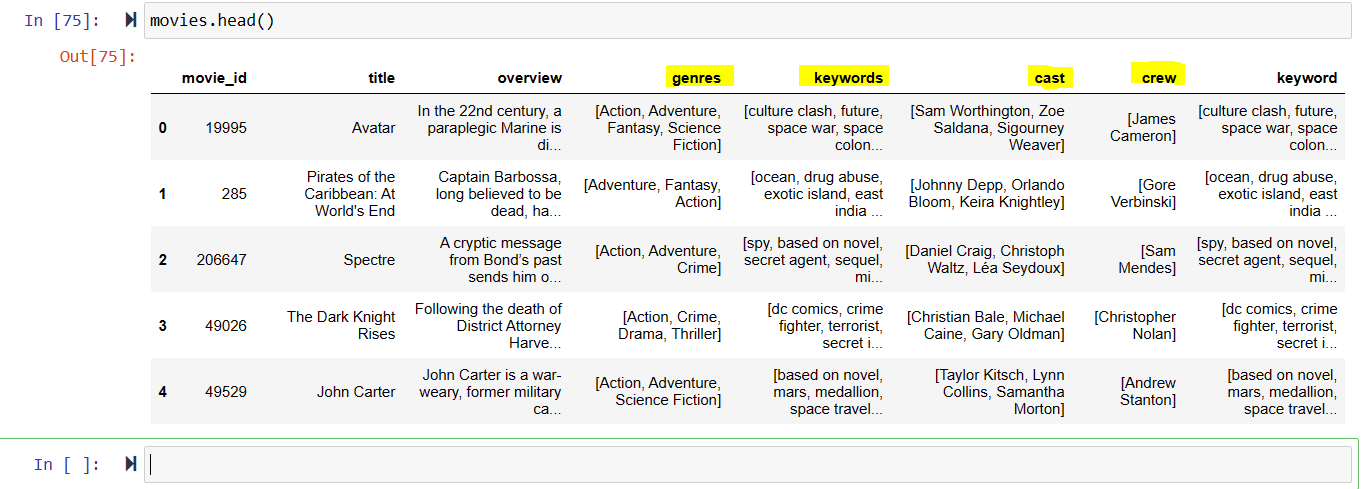
Movies [‘cast’] has a lot of data, we want only 1st three, so we create a function ‘convert3’:



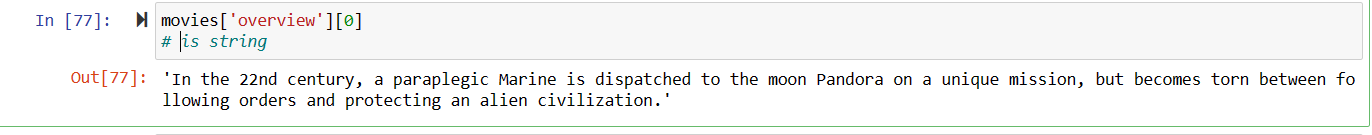
function ‘convert3’:  




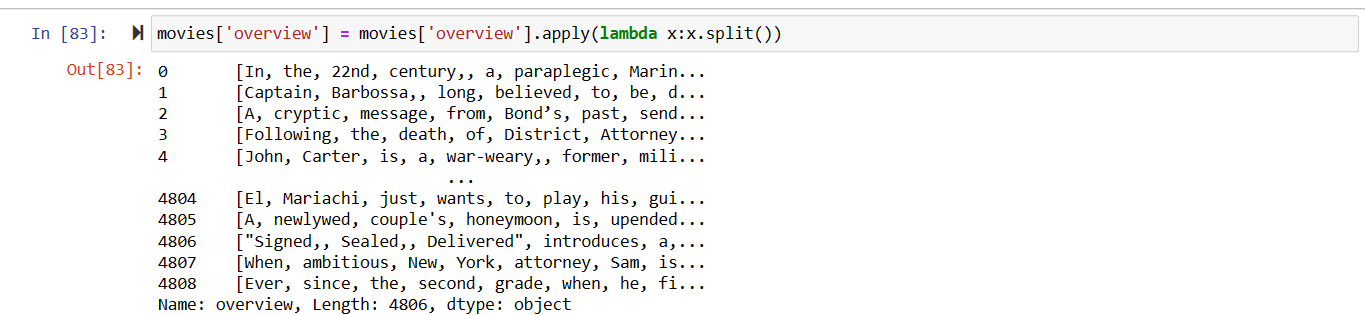
# from movies[‘crew’] we want only the data where ‘job’ is ‘director’:

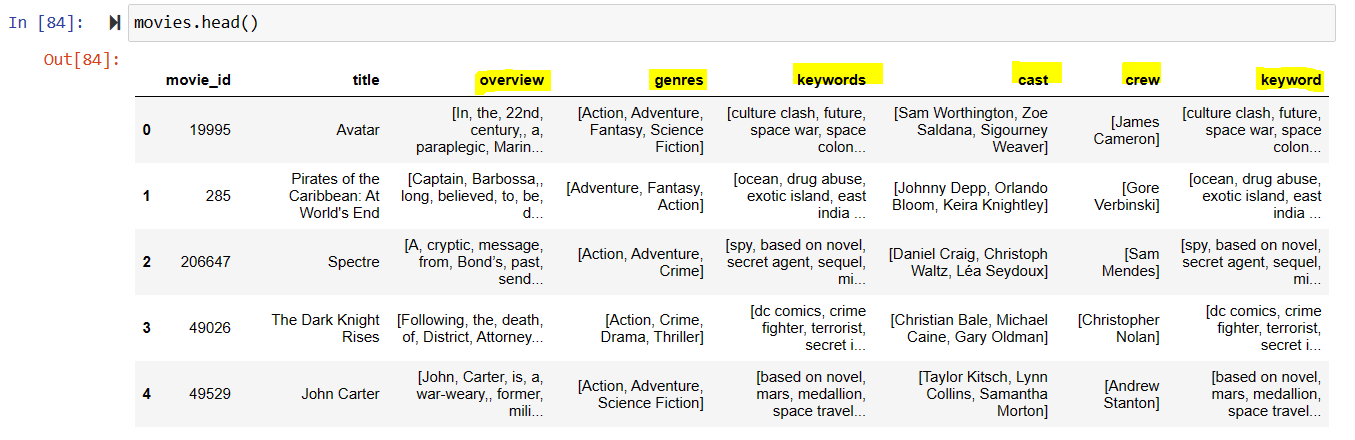


# movies [‘overview’] is string



# converts movie[‘overview’] to list

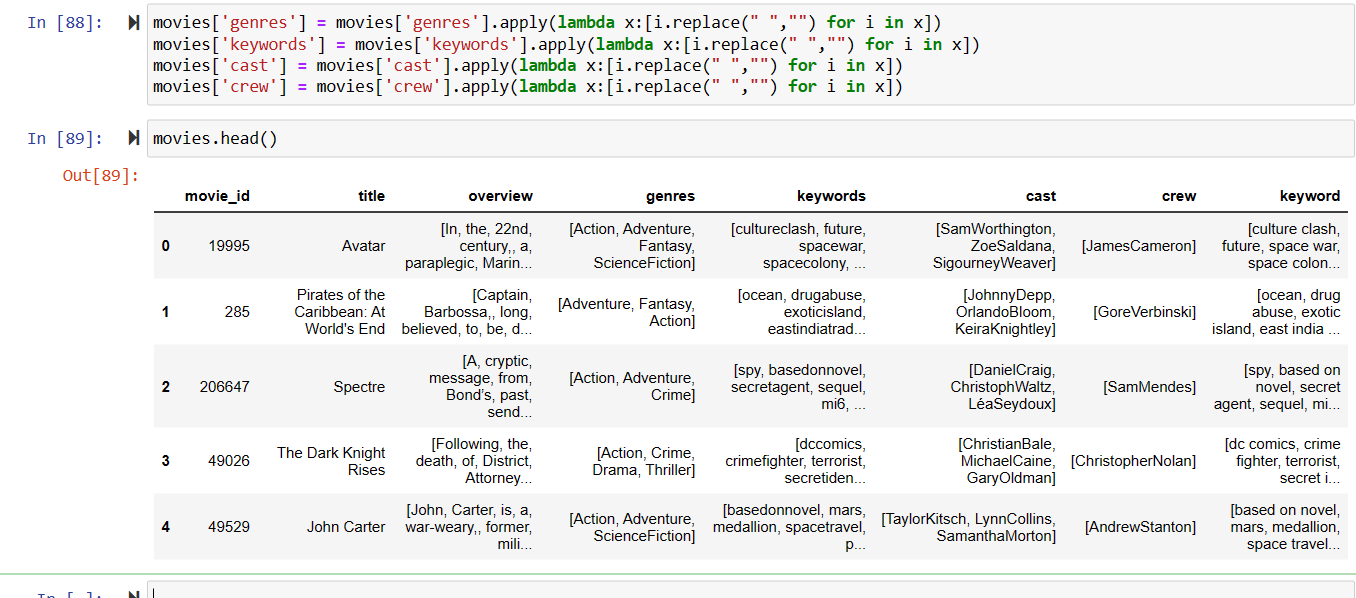


# columns in list form:  


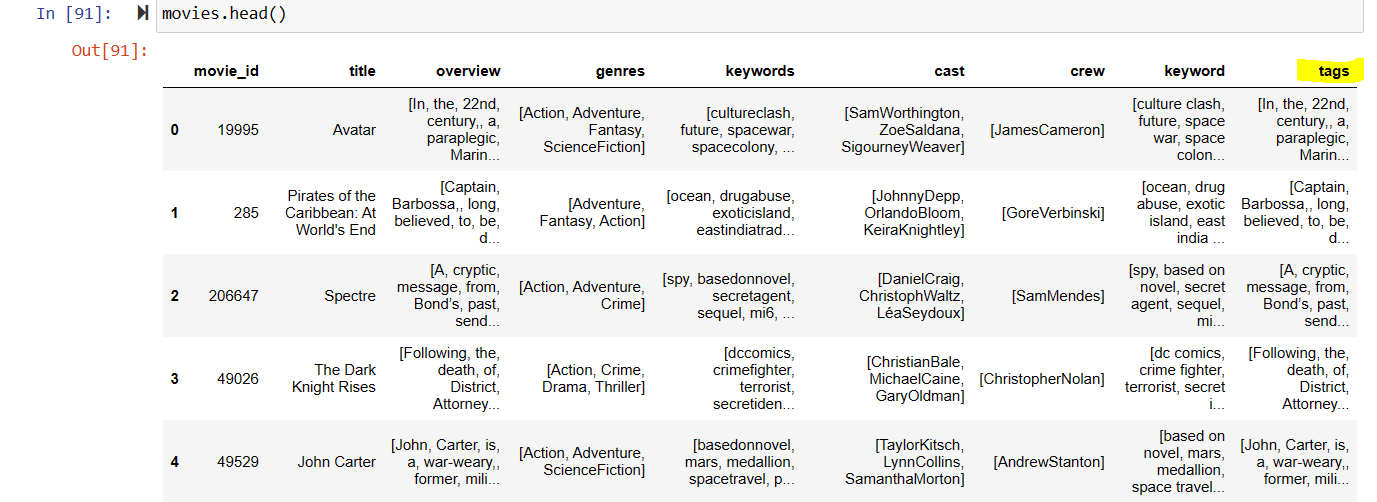
# After converting all columns in lists format, we’ll concatenate them and make a big list and convert that list in a string paragraph and that paragraph will become the ‘tags’ columns.

# The problem:

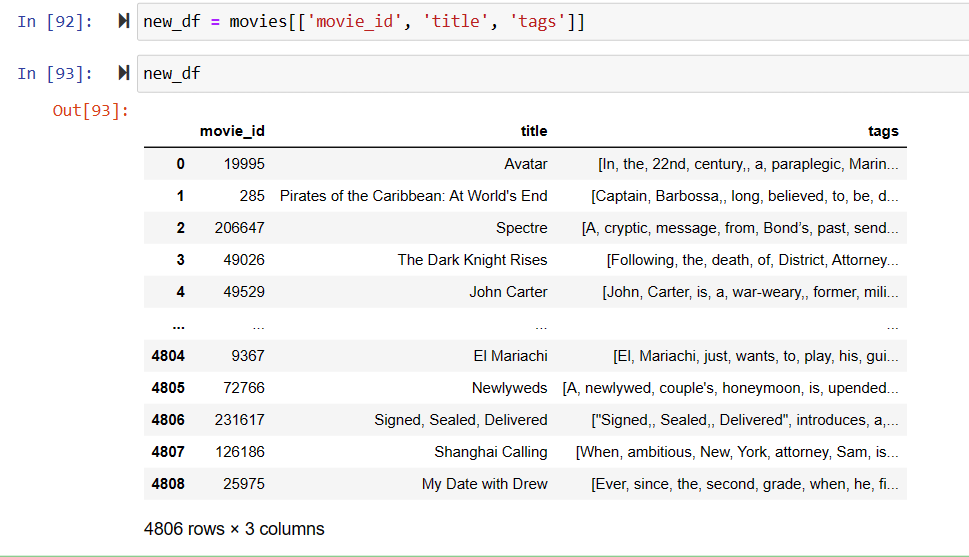
In columns – keywords, genres, cast, crew; we’ll have to apply a transformation such that it removes space between 2 words because if you’re looking for a movie with name ‘Sam’ there are 2 ‘Sam’ and there is a possibility that if I’m looking for “Sam Worthington’s” movie I might get “Sam Mendes’” movie recommended.

# Removing spaces “ “ from ‘overview’, ‘genres’, ‘keywords’, ‘cast’, ‘crew’ columns:  


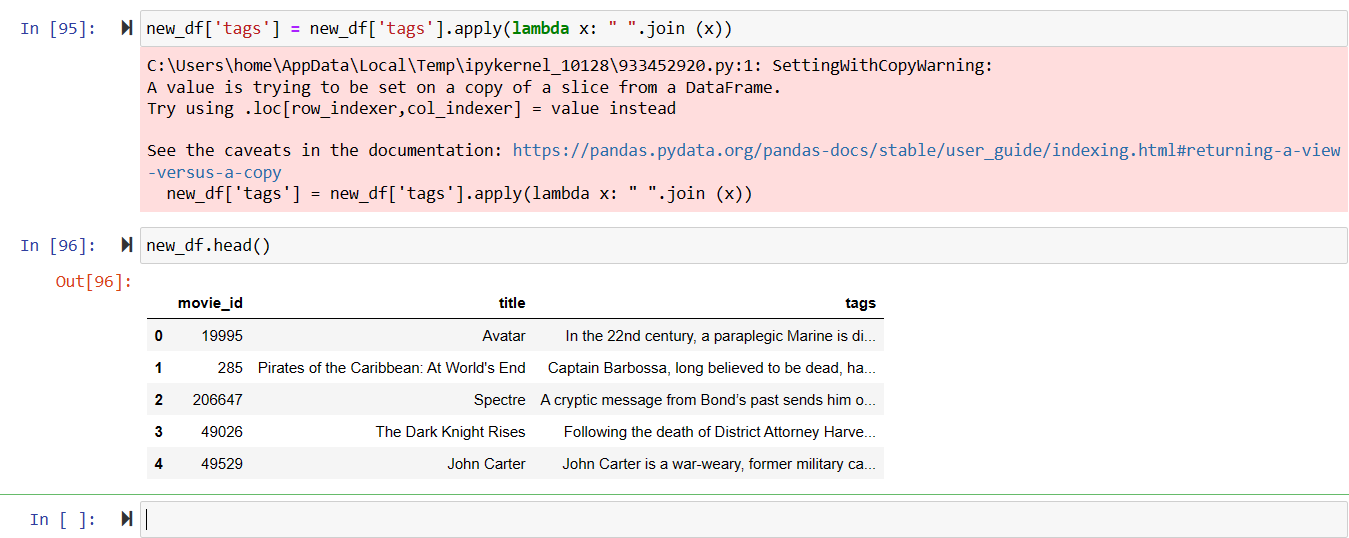
# Now that spaces are removed, we are going to concatenate all these columns in a single column named ‘tags’:

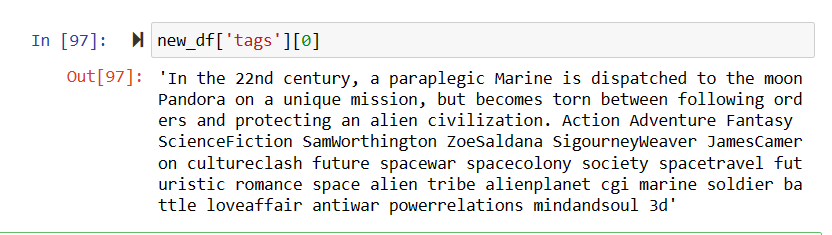


* Since we have concatenated all the columns (overview, genres, keywords, cast, crew, keyword) in ‘tags’ column we can delete all these individual columns and saving these 3 columns (movie\_id, title, tags) in a new dataframe names ‘new\_df’:



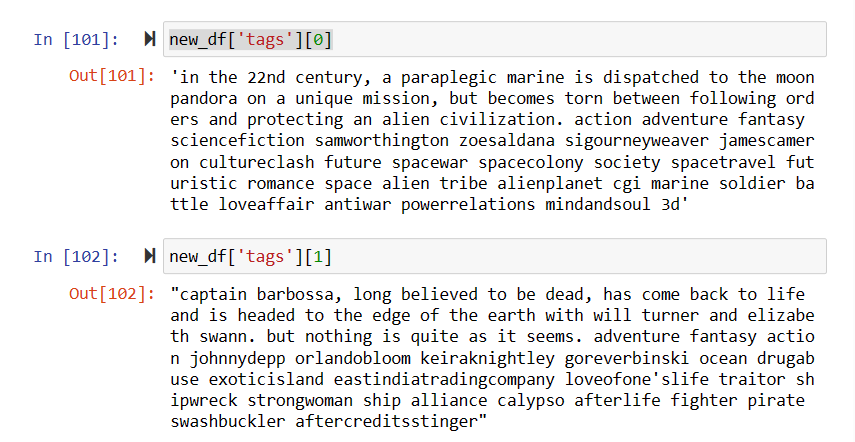
Joining elements in ‘tags’ by “ “ space





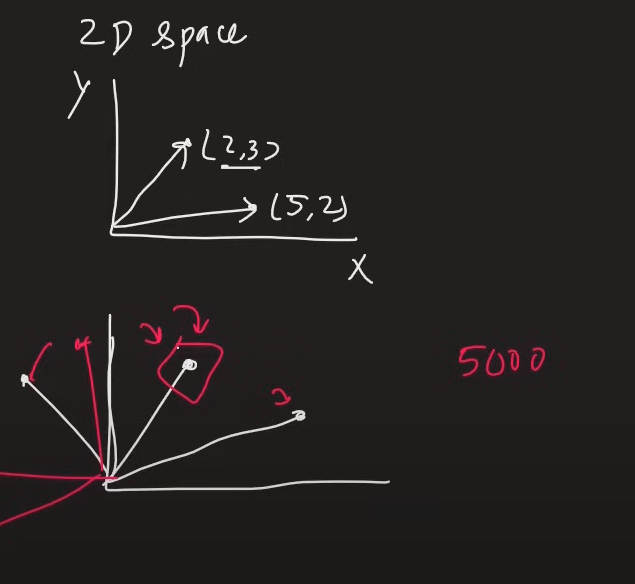
**Problem statement:**  
We have to build a website where the user will give name of a movie and we have to suggest 5 movies related to the user movie. Similarity will be determined on the basis of tags.

Now how can we find the similarities between 5000 movies?



We can’t be comparing each word to find similar words for each movie, so we use vectorization.

Where we’ll display the tags column of the dataset in vector form and to see if movies are similar, we can use vectorization to check, the closer a point is to another the more similar they are.



# Text Vectorization

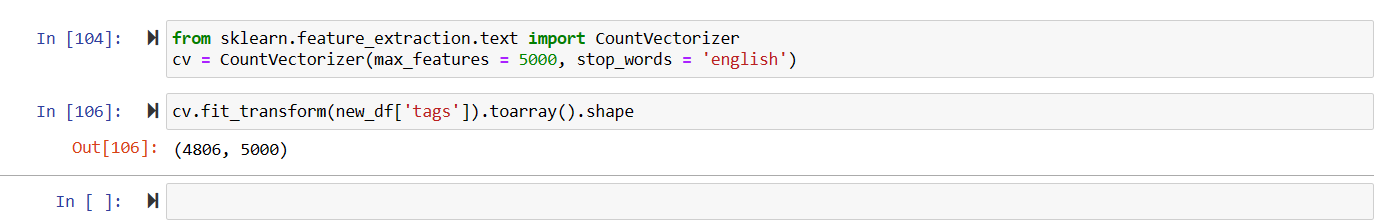
**Text Vectorization** is the process of converting text into numerical representation. Here are some popular methods to accomplish text vectorization:

* Binary Term Frequency
* **Bag of Words (BoW) Term Frequency**
* (L1) Normalized Term Frequency
* (L2) Normalized TF-IDF
* Word2Vec

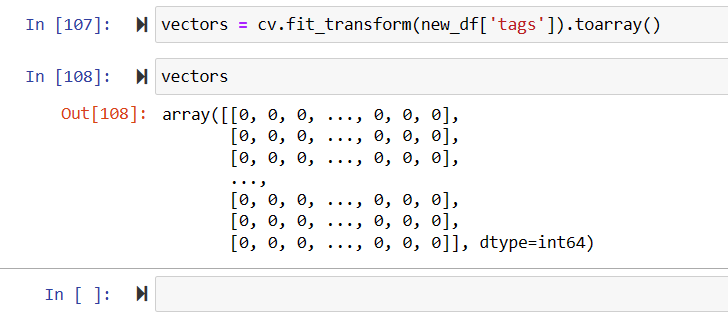
**Bag of Words (BoW) Term Frequency**

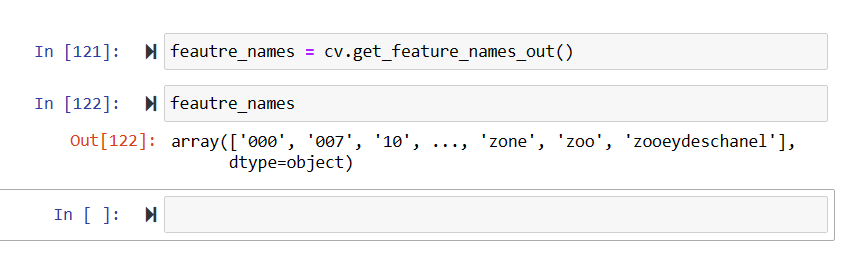
In BoW we combine tags of all the movies which results in a huge string (say ‘large\_text’ variable). Now from that string we ant the 5000 most used words and will calculate which words are the most used and calculate its frequency and extract them. (Let the 5000 words e - w1, w2, w3, w4, …w5000) For every word in ‘tags’ of the 5000 movies we’ll check number of times these words occur. For the process of vectorization, we won’t consider stop words.

Using scikit learn:



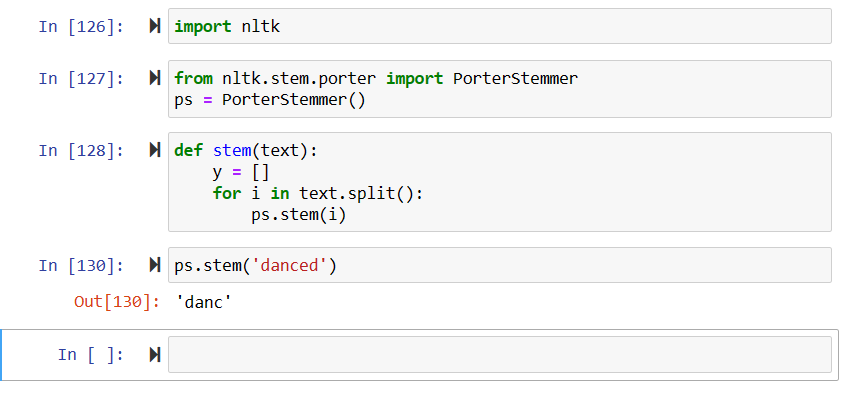
0 if the word doesn’t exist, else 1



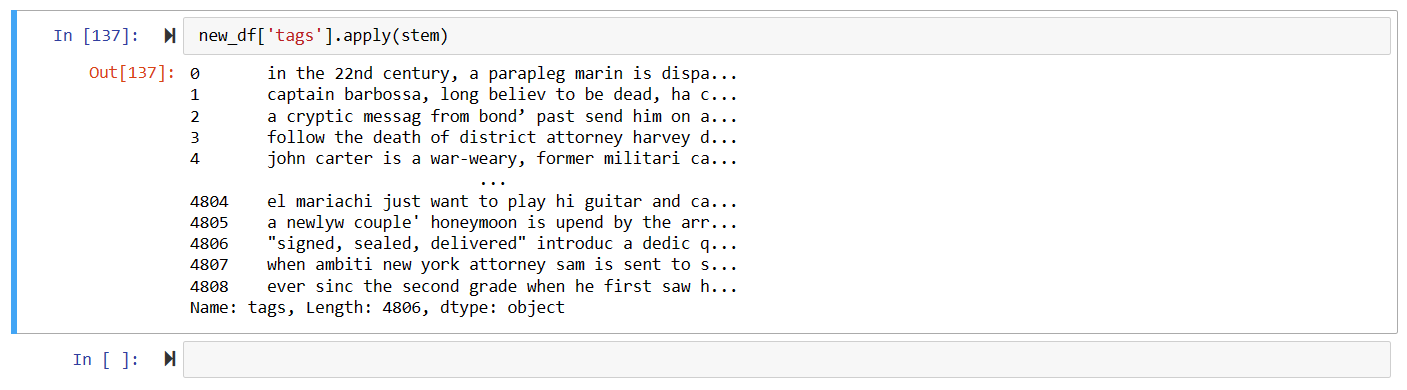


Now some words here might be like: dance, danced, dancing; with same meaning but just different tense, so we’ll replace these by ‘dance’ (here) and then remove copies for that we’ll use ‘nltk’ library.

Using .stem() function of nltk library to get the root from given word



This is how our dataset looks like when taken out only the root word:

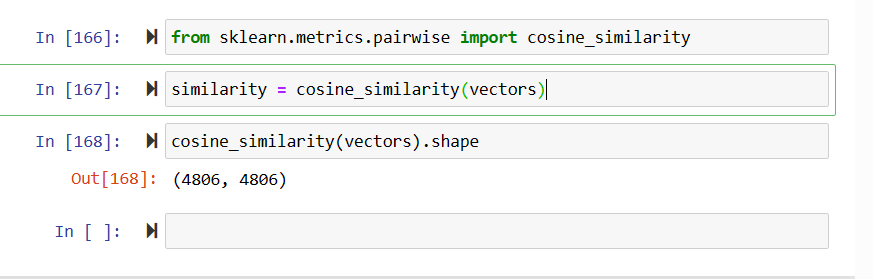


We have 4806 movies in all. Which is same number as vectors and each vector has 5000 columns.

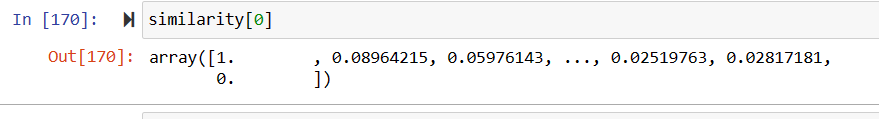
We have 5000-dimensional coordinate space, every movie there is a vector. And we need to find distance between each. More distance, less similarity.

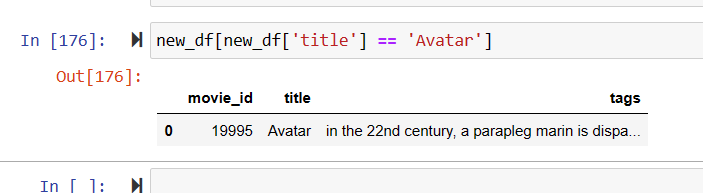
To measure we’ll not find ‘Euclidean distance’ (tip to tip distance) we’ll find ‘cosine distance’ (angle between them).

(4806, 4806) distance between 1 to 4806 points, 2 to 4806 points, … 4806 to 4806 points



Distance of 1st movie to all 4806 points:



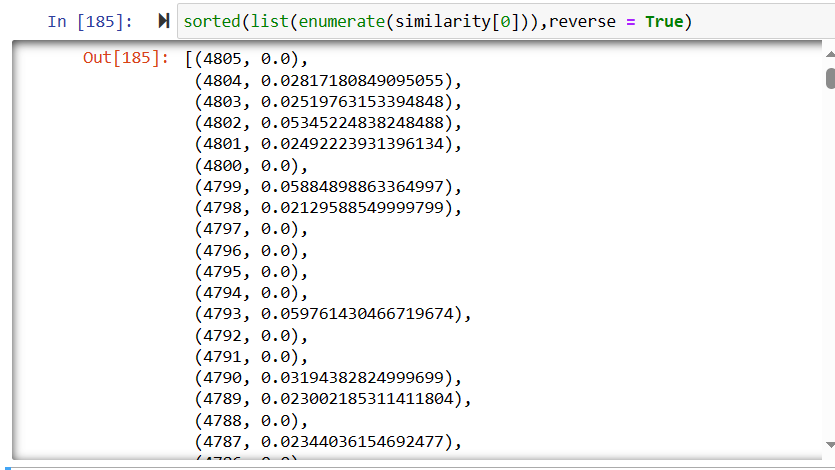


If we sort these we’ll lose the index number i.e., eg distance between 1 to 1, 1 to 2, 1 to 4608 will all get jumbled.

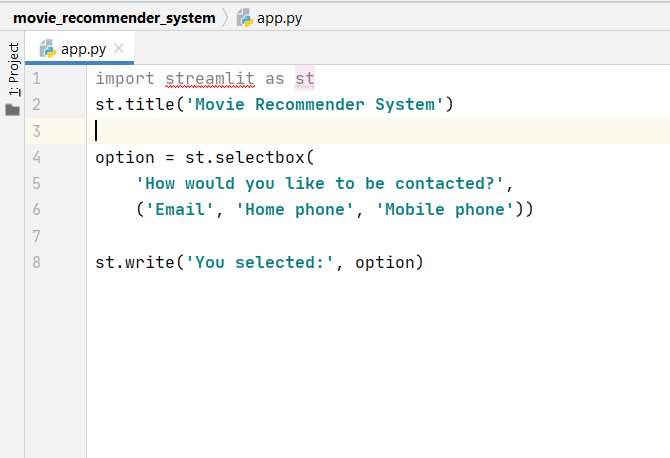
Hence we use enumerate to get index number with it:



Sorts on basis of index number, we need to change that to sort on basis of 2nd element :



Open pycharm and make a project ‘movie recommendation system’ and file ‘app.py’

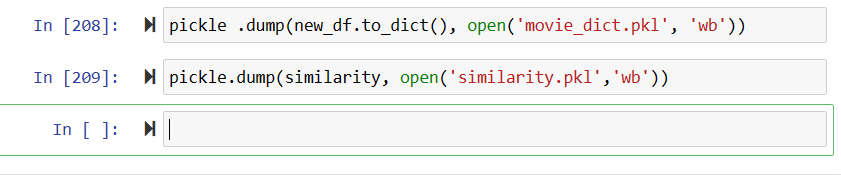


This is how the output will look like:



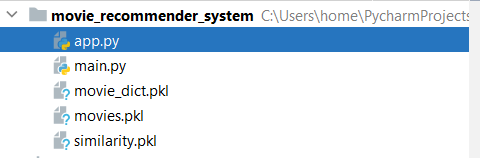
To send file from jupyter notebook to pycharm we use ‘pickle’ library.

Rather than sending the whole data frame we send the dictionary





Copy them in app.py folder



* To get posters for movies we’re going to use tmdb API

[https://api.themoviedb.org/3/keyword/{keyword\_id}/movies](https://api.themoviedb.org/3/keyword/%7bkeyword_id%7d/movies)

**References**:  
<https://medium.com/mlearning-ai/what-are-the-types-of-recommendation-systems-3487cbafa7c9>

Dataset: <https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata>

<https://towardsdatascience.com/getting-started-with-text-vectorization-2f2efbec6685>