```
DATA_FILE_ROOT_PATH='./data/'
```

```
import pandas as pd
import time
```

```
path_movies              = DATA_FILE_ROOT_PATH + 'data_movies_active.csv'  #change dir to your project folder
path_movies_clusters     = DATA_FILE_ROOT_PATH + 'movies_analysis_format_wCluster.csv'  #change dir to your project folder

#added "low memory = False" to avoid a dtype error on loading csv. Seems like it scans a part of the file and may not have i
data_movies = pd.read_csv(path_movies,low_memory=False)
movies_analysis_format_wCluster = pd.read_csv(path_movies_clusters, low_memory=False)
```

```
data_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5528 entries, 0 to 5527
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Unnamed: 0             5528 non-null   int64
 1   imdb_title_id          5528 non-null   object
 2   title                  5528 non-null   object
 3   original_title         5528 non-null   object
 4   year                   5528 non-null   int64
 5   date_published         5528 non-null   object
 6   genre                  5528 non-null   object
 7   duration               5528 non-null   int64
 8   country                5528 non-null   object
 9   language               5528 non-null   object
 10  director               5524 non-null   object
 11  writer                 5494 non-null   object
 12  production_company     5360 non-null   object
 13  actors                 5523 non-null   object
 14  description            5512 non-null   object
 15  avg_vote               5528 non-null   float64
 16  votes                  5528 non-null   int64
 17  budget                 2293 non-null   object
 18  usa_gross_income       2050 non-null   object
 19  worlwide_gross_income  2192 non-null   object 
 20  metascore              1831 non-null   float64
 21  reviews_from_users     5479 non-null   float64
 22  reviews_from_critics   5235 non-null   float64
dtypes: float64(4), int64(4), object(15)
memory usage: 993.4+ KB
```

```
movies_analysis_format_wCluster.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33577 entries, 0 to 33576
Data columns (total 28 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Unnamed: 0             33577 non-null  int64  
 1   imdb_title_id          33577 non-null  object 
 2   title                  33577 non-null  object 
 3   original_title         33577 non-null  object 
 4   year                   33577 non-null  int64  
 5   date_published         33577 non-null  object 
 6   genre                  33577 non-null  object 
 7   duration               33577 non-null  int64  
 8   country                33577 non-null  object 
 9   language               33577 non-null  object 
 10  director               33577 non-null  object 
 11  writer                 33445 non-null  object 
 12  production_company     32924 non-null  object 
 13  actors                 33577 non-null  object 
 14  description            33490 non-null  object 
 15  avg_vote               33577 non-null  float64
 16  votes                  33577 non-null  int64  
 17  budget                 16447 non-null  object 
 18  usa_gross_income       15203 non-null  object 
 19  worlwide_gross_income  16854 non-null  object 
 20  metascore              14096 non-null  float64
 21  reviews_from_users     33335 non-null  float64
 22  reviews_from_critics   32320 non-null  float64
 23  imdb_actor_id          33577 non-null  object 
 24  imdb_director_id       33577 non-null  object 
 25  Genre                  33577 non-null  object 
 26  Country                33577 non-null  object 
 27  Cluster                33577 non-null  object 
```

```
dtypes: float64(4), int64(4), object(20)
memory usage: 7.2+ MB
```

Recommender Functions

```
#Constants for recommender functionality
UNIQUE_TS_COLUMN_NAME='uniqueTempCol'

#if using the full set ~472K rows it crashes with out of memory error
QUICK_TEST_RECOMMENDER=True

WEIGHT_CLUSTER=1
WEIGHT_CLUSTER_GENRE=3
WEIGHT_CLUSTER_DIRECTOR=10
WEIGHT_CLUSTER_DIRECTOR_ACTOR=10
WEIGHT_CLUSTER_GENRE_ACTOR=10
WEIGHT_CLUSTER_GENRE_DIRECTOR=10

NUM_RECOMMENDED_MOVIES=10


def get_movie_details_for_analysis(imdb_title_id):
  #get record of movie details
  return movies_analysis_format_wCluster[movies_analysis_format_wCluster['imdb_title_id']==imdb_title_id]


#returns dataframe with top 10 movies based on rank
#lots of duplicate ranks so we need a tie breaker or another filter

def getTopForCluster(Cluster):

  df = movies_analysis_format_wCluster

  df = df[df.Cluster.isin([Cluster])]
  df = df[['Cluster', 'avg_vote']]
  #display(df.shape)
  df = df.drop_duplicates()
  #display(df.shape)
  df = df.sort_values(by=['avg_vote'], ascending=False )
  df = df.head(WEIGHT_CLUSTER)  #now have the top 10 ranks.

  df = pd.merge(df, movies_analysis_format_wCluster, how="inner", on=['avg_vote', 'Cluster'])
  df[UNIQUE_TS_COLUMN_NAME] = time.time()

  return df




#returns dataframe with top 10 movies based on rank
#lots of duplicate ranks so we need a tie breaker or another filter

def getTopForClusterGenre(Cluster,Genre):

  df = movies_analysis_format_wCluster

  df = df[df.Cluster.isin([Cluster])]
  df = df[df.Genre.isin([Genre])]
  df = df[['Cluster', 'avg_vote','Genre']]
  #display(df.shape)
  df = df.drop_duplicates()
  #display(df.shape)
  df = df.sort_values(by=['avg_vote'], ascending=False )
  df = df.head(WEIGHT_CLUSTER_GENRE)  #now have the top 10 ranks.

  df = pd.merge(df, movies_analysis_format_wCluster, how="inner", on=['avg_vote', 'Cluster', 'Genre'])
  df[UNIQUE_TS_COLUMN_NAME] = time.time()

  return df
```

```
#returns dataframe with top 10 movies based on rank
#lots of duplicate ranks so we need a tie breaker or another filter

def getTopForClusterByDirector(Cluster, Director):
  df = movies_analysis_format_wCluster

  df = df[df.Cluster.isin([Cluster])]
  df = df[df.imdb_director_id.isin([Director])]

  df = df[['Cluster', 'avg_vote', 'imdb_director_id']]
  #display(df.shape)

  df = df.drop_duplicates()
  #display(df.shape)
  df = df.sort_values(by=['avg_vote'], ascending=False )
  df = df.head(WEIGHT_CLUSTER_DIRECTOR)  #now have the top 10 ranks.

  df = pd.merge(df, movies_analysis_format_wCluster, how="inner", on=['avg_vote', 'Cluster', 'imdb_director_id'])
  df[UNIQUE_TS_COLUMN_NAME] = time.time()

  return df


#returns dataframe with top 10 movies based on rank
#lots of duplicate ranks so we need a tie breaker or another filter

def getTopForClusterByDirectorActor(Cluster, Director, Actor):
  df = movies_analysis_format_wCluster

  df = df[df.Cluster.isin([Cluster])]
  df = df[df.imdb_director_id.isin([Director])]
  df = df[df.imdb_actor_id.isin([Actor])]
  df = df[['Cluster', 'avg_vote', 'imdb_director_id', 'imdb_actor_id']]
  #display(df.shape)
  df = df.drop_duplicates()
  #display(df.shape)
  df = df.sort_values(by=['avg_vote'], ascending=False )
  df = df.head(WEIGHT_CLUSTER_DIRECTOR_ACTOR)  #now have the top 10 ranks.
```

```
    df = pd.merge(df, movies_analysis_format_wCluster, how="inner", on=['avg_vote', 'Cluster', 'imdb_director_id', 'imdb_actor_
    df[UNIQUE_TS_COLUMN_NAME] = time.time()

    return df

  #returns dataframe with top 10 movies based on rank
  #lots of duplicate ranks so we need a tie breaker or another filter

  def getTopForClusterByGenreActor(Cluster, Genre, Actor):
    df = movies_analysis_format_wCluster

    df = df[df.Cluster.isin([Cluster])]
    df = df[df.Genre.isin([Genre])]
    df = df[df.imdb_actor_id.isin([Actor])]
    df = df[['Cluster', 'avg_vote', 'Genre', 'imdb_actor_id']]
    #display(df.shape)
    df = df.drop_duplicates()
    #display(df.shape)
    df = df.sort_values(by=['avg_vote'], ascending=False )
    df = df.head(WEIGHT_CLUSTER_GENRE_ACTOR)  #now have the top 10 ranks.

    df = pd.merge(df, movies_analysis_format_wCluster, how="inner", on=['avg_vote', 'Cluster', 'Genre', 'imdb_actor_id'])
    df[UNIQUE_TS_COLUMN_NAME] = time.time()

    return df


  #returns dataframe with top 10 movies based on rank
  #lots of duplicate ranks so we need a tie breaker or another filter

  def getTopForClusterByGenreDirector(Cluster, Genre, Director):
    df = movies_analysis_format_wCluster

    df = df[df.Cluster.isin([Cluster])]
    df = df[df.Genre.isin([Genre])]
    df = df[df.imdb_director_id.isin([Director])]
    df = df[['Cluster', 'avg_vote', 'Genre', 'imdb_director_id']]
    #display(df.shape)
    df = df.drop_duplicates()
    #display(df.shape)
    df = df.sort_values(by=['avg_vote'], ascending=False )
    df = df.head(WEIGHT_CLUSTER_GENRE_DIRECTOR)  #now have the top 10 ranks.

    df = pd.merge(df, movies_analysis_format_wCluster, how="inner", on=['avg_vote', 'Cluster', 'Genre', 'imdb_director_id'])
    df[UNIQUE_TS_COLUMN_NAME] = time.time()

    return df


  def magicRecommender(dfInput,orig_imdb_title_id):
    dfLarge = dfInput

    dfLarge = dfLarge[~dfLarge.imdb_title_id.isin([orig_imdb_title_id])]

    #remove the move we were called with
    dfLarge = dfLarge.groupby(['imdb_title_id']).size().to_frame('weighting')

    dfLarge = dfLarge.sort_values(by=['weighting'], ascending=False )
    dfLarge = dfLarge.head(NUM_RECOMMENDED_MOVIES)

    dfLarge = pd.merge(dfLarge, data_movies, how="inner", on=['imdb_title_id'])
    #dfLarge.head(5)
    dfLarge = dfLarge[['imdb_title_id', 'weighting','original_title','year', 'genre', 'director', 'actors', 'avg_vote']]
    #dfLarge.head(5)
    return dfLarge



  #Three main dataFrames needed
  #all movies in model format (no )
  #Dataframe: movie_analysis_format
  #
  #all movies in model format with cluster (for separation)
  #Need to run "result" through the model and save it
  #Dataframe: movie_analysis_format_clustered
  #
  #all movies in original format
```

```
#data_movies (filter for content)
#data_movies (remove bad record)
#Dataframe: data_movies

#main function needs to receive a movie ID as input. return as dataframe
def getRecommendations(imdb_title_id):

  #get dataframe of movie rows from data_analysis
  tMovieForAnalysis = get_movie_details_for_analysis(imdb_title_id)
  if (len(tMovieForAnalysis.index)==0):
    display("Could not find movie in file")

  tMovieForAnalysis.reset_index(inplace=True)
  #reqMovie.info()

  #loop through all rows in the Movie cluster set
  for index, row in tMovieForAnalysis.iterrows():
    #display(index)

    #could be in multiple clusters

    #get top 10 movies by actor/genre in cluster
    if(index == 0):
      #display("First loop, creating dataframe")
      retMovies = getTopForCluster(row['Cluster'])

    else:
      #display("Not first loop, appending dataframe")
      retMovies = retMovies.append(getTopForCluster(row['Cluster']))


    retMovies = retMovies.append(getTopForClusterByGenreActor(row['Cluster'],
                                          row['Genre'],
                                          row['imdb_actor_id']))



    #get top by Genre by Cluster
    retMovies = retMovies.append(
        getTopForClusterGenre(row['Cluster'],row['Genre']))

    #get top by Director in Clusters
    retMovies = retMovies.append(
        getTopForClusterByDirector(row['Cluster'],
                                         row['imdb_director_id']))

    #Get top by actor/director in cluster
    retMovies = retMovies.append(
        getTopForClusterByDirectorActor(row['Cluster'],
                                              row['imdb_director_id'],
                                              row['imdb_actor_id']))


    #get top by genre/director
    retMovies = retMovies.append(
        getTopForClusterByGenreDirector(row['Cluster'],
                                              row['Genre'],
                                              row['imdb_director_id']))

  #group em all and do some magic
  retMovies = magicRecommender(retMovies,imdb_title_id)

  return retMovies
```

```
#Need to know some of the movies that are in this dataset for testing
moviesToTest = data_movies.sample(n=10)
moviesToTest.shape
moviesToTest = moviesToTest[['imdb_title_id',
                             'original_title',
                             'year',
                             'genre',
                             'director',
                             'actors',
                             'avg_vote'
```

```
                              ]]
moviesToTest = moviesToTest.sort_values(by=['avg_vote'], ascending=False )
moviesToTest.head(10)
```

	imdb_title_id	original_title	year	genre	director	actors	avg_vot
4463	tt0076759	Star Wars	1977	Action, Adventure, Fantasy	George Lucas	Mark Hamill, Harrison Ford, Carrie Fisher, Pet...	8
1345	tt0080360	Altered States	1980	Horror, Sci-Fi, Thriller	Ken Russell	William Hurt, Blair Brown, Bob Balaban, Charle...	6
						Red	

```
recommendedMovies = getRecommendations('tt0217756')
#display(recommendedMovies.info())
recommendedMovies.head(10)
```

	imdb_title_id	weighting	original_title	year	genre	director	actors	avg_vote
0	tt0765476	16	Meet Dave	2008	Adventure, Comedy, Family	Brian Robbins	Eddie Murphy, Elizabeth Banks, Gabrielle Union...	5.0
1	tt0412080	12	The World's Fastest Indian	2005	Biography, Drama, Sport	Roger Donaldson	Anthony Hopkins, Iain Rea, Tessa Mitchell, Aar...	7.8
2	tt0091217	8	Hoosiers	1986	Drama, Sport	David Anspaugh	Gene Hackman, Barbara Hershey, Dennis Hopper, ...	7.5
3	tt10218912	8	As I Am	2019	Drama, Fantasy, Romance	Anthony Bawn	Andre Myers, Jerimiyah Dunbar, Rodney Chester,...	9.3
4	tt2278388	8	The Grand Budapest Hotel	2014	Adventure, Comedy, Crime	Wes Anderson	Ralph Fiennes, F. Murray Abraham, Mathieu Amal...	8.1
5	tt8832158	8	Poeta	2019	Comedy, Drama, Romance	Jeral Clyde II	Lauren Amelia Arouni, Jabari Hollis, Deanna Th...	8.3
6	tt0028772	4	A Day at the Races	1937	Comedy,	Sam Wood	Groucho Marx, Chico Marx,	7.6