In [629]:
```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

In [630]:
```
#Initialize a TFIDF Vectroizer
tfidf = TfidfVectorizer(stop_words='english')
```

In [631]:
```
movies=pd.read_csv("IMDBdata_MainData.csv")
```

In [632]:
```
movies.columns
```

Out[632]:
```
Index(['Title', 'Year', 'Rated', 'Released', 'Runtime', 'Genre', '
Director',
       'Writer', 'Actors', 'Plot', 'Language', 'Country', 'Awards'
, 'Poster',
       'Ratings.Source', 'Ratings.Value', 'Metascore', 'imdbRating
',
       'imdbVotes', 'imdbID', 'Type', 'DVD', 'BoxOffice', 'Product
ion',
       'Website', 'Response', 'tomatoURL'],
      dtype='object')
```

In [633]:
```
movies=movies[['Title','Plot','Director','Genre']]
```

In [634]:
```
movies.head()
```

Out[634]:

	Title	Plot	Director	Genre
0	Code Name: K.O.Z.	A look at the 17-25 December 2013 corruption s...	Celal Çimen	Crime, Mystery
1	Saving Christmas	Kirk is enjoying the annual Christmas party ex...	Darren Doane	Comedy, Family
2	Superbabies: Baby Geniuses 2	A group of smart-talking toddlers find themsel...	Bob Clark	Comedy, Family, Sci-Fi
3	Daniel der Zauberer	Evil assassins want to kill Daniel Kublbock, t...	Ulli Lommel	Comedy, Crime, Fantasy
4	Manos: The Hands of Fate	A family gets lost on the road and stumbles up...	Harold P. Warren	Horror

In [635]:
```
movies['Plot'].fillna(" ",inplace=True)
```

In [636]:
```
#Create a map of movie tilte to its index
AllIndex = pd.Series(movies.index, index=movies['Title']).drop_dupl
icates()
```

In [637]:
```
#Get the TFIDF matrix fittend on movie's Plot
matrix = tfidf.fit_transform(movies['Plot'])
```

In [638]:
```
#Use cosine similarity measure to calculate the similarity between
the movies
#Since you have used the TF-IDF vectorizer, calculating the dot pro
duct will directly give you
#the cosine similarity score
cosine_similarity = linear_kernel(matrix,matrix)
```

In [639]:
```
#Function to give top 10 movie recommendations

def recommendations(title,cs=cosine_similarity):
    """
    :param: title - title of the movie to find similar movies
    :param: cs - Cosine similarty matrix
    """
    index=AllIndex[title]

    #Get all similarity scores for the given movie
    scores = list(enumerate(cs[index]))
    scores = sorted(scores, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar movies
    scores = scores[1:11]

    # Get the movie indices for top 10
    movie_indices = [i[0] for i in scores]

    return movies['Title'].iloc[movie_indices]
recommendations('The Dark Knight Rises')
```

Out[639]:
```
1960                              Batman
103                    The Dark Knight
4674                               Yes
1218                         The Siege
2239                           Godsend
2458                         Elizabeth
3749    The Wind That Shakes the Barley
1064                     Batman Returns
2122              George and the Dragon
3249                           Flipped
Name: Title, dtype: object
```

In [640]:
```
recommendations('The Godfather: Part II')
```

Out[640]:
```
898                    The Aviator
1486     The Godfather: Part III
4377                     The Purge
1556               Absolute Power
4348                 The Betrayed
2172            Midnight in Paris
4245                      Boyhood
1429    The Pursuit of Happyness
2247                Dead Man Down
1013         The Nutcracker in 3D
Name: Title, dtype: object
```