Title.basics.tsv

- This dataset column titleType contains following values

```
df.titleType.value_counts()
```

```
tvEpisode       3953453
short            671435
movie            514140
video            225403
tvSeries         161535
tvMovie          126081
tvMiniSeries      25283
videoGame         23157
tvSpecial         16843
tvShort            9111
Name: titleType, dtype: int64
```

-
- We clubbed above values into episode, movie and series.

Movie dataframe

```
dfMovie = df[(df["titleType"] == "short") | (df["titleType"] == "tvShort") | (df["titleType"] == "movie")
             | (df["titleType"] == "tvMovie")]
dfMovie["titleType"] = "movie"
```

...

Series dataset

```
dfSeries = df[(df["titleType"] == "tvSeries") | (df["titleType"] == "tvMiniSeries")]
dfSeries["titleType"] = "series"
```

...

Episode dataset

```
dfEpisode = df[(df["titleType"]== "tvEpisode")]
dfEpisode['titleType'] = 'episode'
```

...

Merge episode, series, movie

```
dfBasicsNew = pd.concat([dfMovie, dfSeries, dfEpisode], ignore_index=True)
dfBasicsNew.head()
```

-
- So there are now 3 titleType values present in dataset

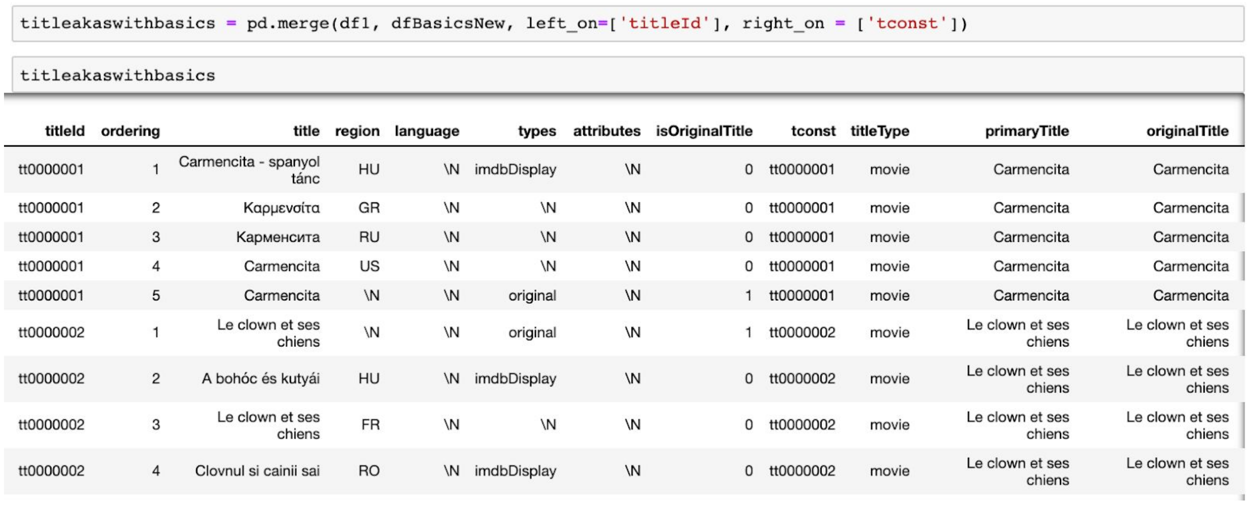
```
dfBasicsNew.titleType.value_counts()
```

```
episode    3953453
movie      1320767
series      186818
Name: titleType, dtype: int64
```

-
- After that we merged this dataset with **ratings dataset** such that titles will get rating and votes count information.

Title.akas.tsv

- We merged this dataset with Title.basics.tsv so that we can get information of titles according to different regions and their languages of each movie, series and episode



```
titleakaswithbasics = pd.merge(df1, dfBasicsNew, left_on=['titleId'], right_on = ['tconst'])
```

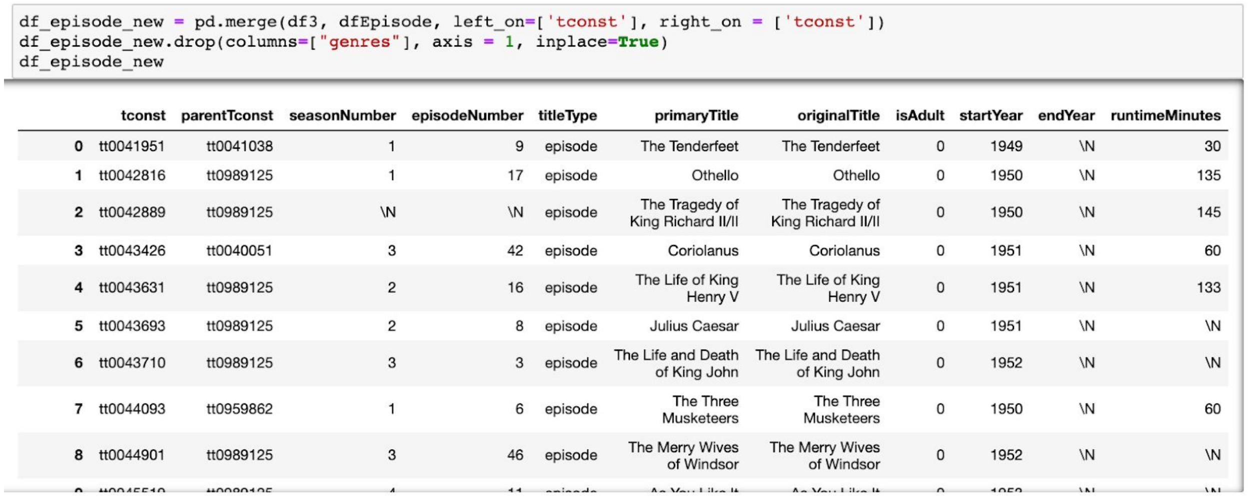
```
titleakaswithbasics
```

titleId	ordering	title	region	language	types	attributes	isOriginalTitle	tconst	titleType	primaryTitle	originalTitle
tt0000001	1	Carmencita - spanyol tánc	HU	\N	imdbDisplay	\N	0	tt0000001	movie	Carmencita	Carmencita
tt0000001	2	Καρμενσίτα	GR	\N	\N	\N	0	tt0000001	movie	Carmencita	Carmencita
tt0000001	3	Карменсита	RU	\N	\N	\N	0	tt0000001	movie	Carmencita	Carmencita
tt0000001	4	Carmencita	US	\N	\N	\N	0	tt0000001	movie	Carmencita	Carmencita
tt0000001	5	Carmencita	\N	\N	original	\N	1	tt0000001	movie	Carmencita	Carmencita
tt0000002	1	Le clown et ses chiens	\N	\N	original	\N	1	tt0000002	movie	Le clown et ses chiens	Le clown et ses chiens
tt0000002	2	A bohóc és kutyái	HU	\N	imdbDisplay	\N	0	tt0000002	movie	Le clown et ses chiens	Le clown et ses chiens
tt0000002	3	Le clown et ses chiens	FR	\N	\N	\N	0	tt0000002	movie	Le clown et ses chiens	Le clown et ses chiens
tt0000002	4	Clovnul si cainii sai	RO	\N	imdbDisplay	\N	0	tt0000002	movie	Le clown et ses chiens	Le clown et ses chiens

-
- Then we segregated this whole dataset by movie, series and episode titleTypes datasets.

Title.episode.tsv

- As we already have information about episodes from title.basics dataset, we merge this dataset with title.basics



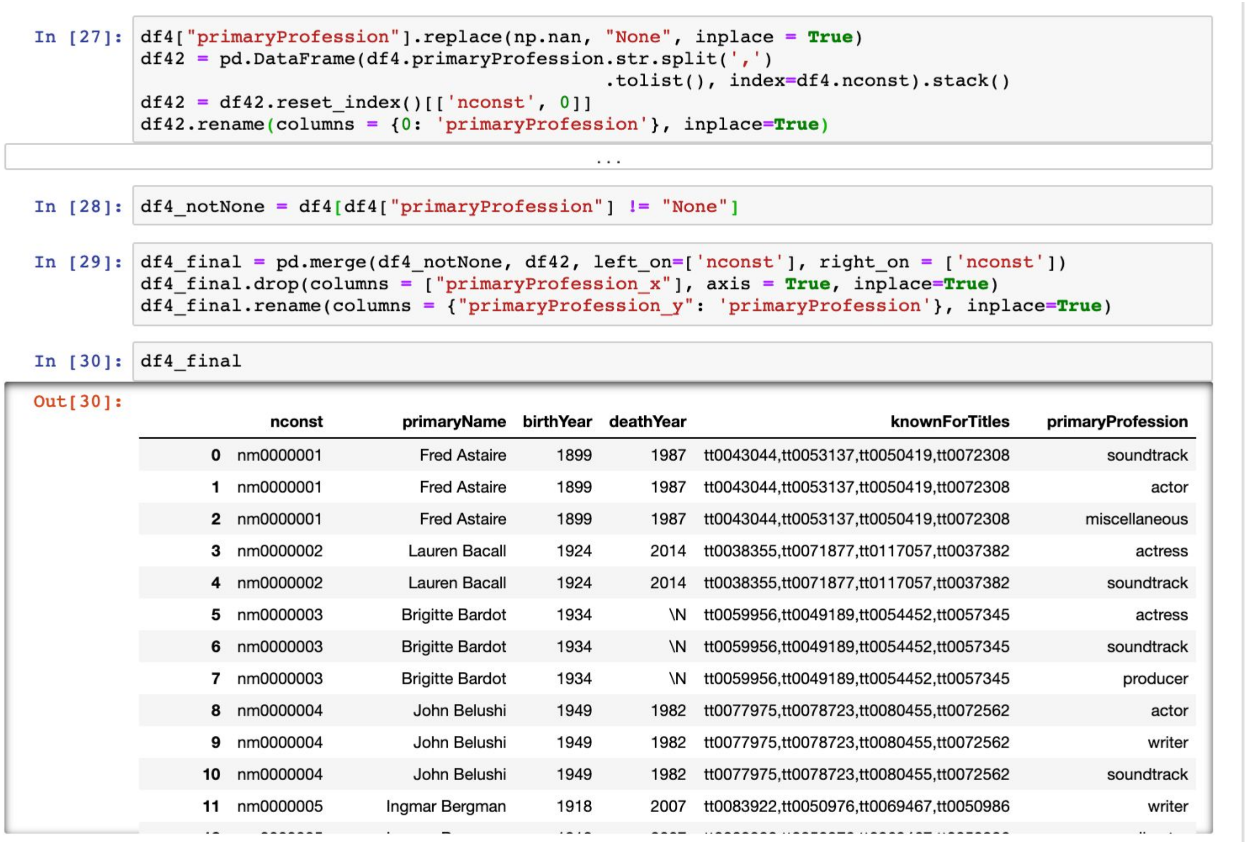
```
df_episode_new = pd.merge(df3, dfEpisode, left_on=['tconst'], right_on = ['tconst'])
df_episode_new.drop(columns=["genres"], axis = 1, inplace=True)
df_episode_new
```

	tconst	parentTconst	seasonNumber	episodeNumber	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes
0	tt0041951	tt0041038	1	9	episode	The Tenderfeet	The Tenderfeet	0	1949	\N	30
1	tt0042816	tt0989125	1	17	episode	Othello	Othello	0	1950	\N	135
2	tt0042889	tt0989125	\N	\N	episode	The Tragedy of King Richard II/II	The Tragedy of King Richard II/II	0	1950	\N	145
3	tt0043426	tt0040051	3	42	episode	Coriolanus	Coriolanus	0	1951	\N	60
4	tt0043631	tt0989125	2	16	episode	The Life of King Henry V	The Life of King Henry V	0	1951	\N	133
5	tt0043693	tt0989125	2	8	episode	Julius Caesar	Julius Caesar	0	1951	\N	\N
6	tt0043710	tt0989125	3	3	episode	The Life and Death of King John	The Life and Death of King John	0	1952	\N	\N
7	tt0044093	tt0959862	1	6	episode	The Three Musketeers	The Three Musketeers	0	1950	\N	60
8	tt0044901	tt0989125	3	46	episode	The Merry Wives of Windsor	The Merry Wives of Windsor	0	1952	\N	\N

-

Name.basics.tsv

- As this dataset contains information of people and their profession in the movie, it also contains multivalued column named primary Profession. We solved this problem by converting those values into equivalent number of rows.

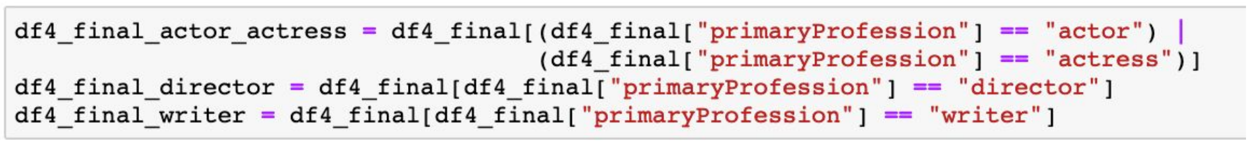


```
In [27]: df4["primaryProfession"].replace(np.nan, "None", inplace = True)
         df42 = pd.DataFrame(df4.primaryProfession.str.split(',')
                                                  .tolist(), index=df4.nconst).stack()
         df42 = df42.reset_index()[['nconst', 0]]
         df42.rename(columns = {0: 'primaryProfession'}, inplace=True)
...

In [28]: df4_notNone = df4[df4["primaryProfession"] != "None"]

In [29]: df4_final = pd.merge(df4_notNone, df42, left_on=['nconst'], right_on = ['nconst'])
         df4_final.drop(columns = ["primaryProfession_x"], axis = True, inplace=True)
         df4_final.rename(columns = {"primaryProfession_y": 'primaryProfession'}, inplace=True)

In [30]: df4_final
```

Out[30]:

	nconst	primaryName	birthYear	deathYear	knownForTitles	primaryProfession
0	nm0000001	Fred Astaire	1899	1987	tt0043044,tt0053137,tt0050419,tt0072308	soundtrack
1	nm0000001	Fred Astaire	1899	1987	tt0043044,tt0053137,tt0050419,tt0072308	actor
2	nm0000001	Fred Astaire	1899	1987	tt0043044,tt0053137,tt0050419,tt0072308	miscellaneous
3	nm0000002	Lauren Bacall	1924	2014	tt0038355,tt0071877,tt0117057,tt0037382	actress
4	nm0000002	Lauren Bacall	1924	2014	tt0038355,tt0071877,tt0117057,tt0037382	soundtrack
5	nm0000003	Brigitte Bardot	1934	\N	tt0059956,tt0049189,tt0054452,tt0057345	actress
6	nm0000003	Brigitte Bardot	1934	\N	tt0059956,tt0049189,tt0054452,tt0057345	soundtrack
7	nm0000003	Brigitte Bardot	1934	\N	tt0059956,tt0049189,tt0054452,tt0057345	producer
8	nm0000004	John Belushi	1949	1982	tt0077975,tt0078723,tt0080455,tt0072562	actor
9	nm0000004	John Belushi	1949	1982	tt0077975,tt0078723,tt0080455,tt0072562	writer
10	nm0000004	John Belushi	1949	1982	tt0077975,tt0078723,tt0080455,tt0072562	soundtrack
11	nm0000005	Ingmar Bergman	1918	2007	tt0083922,tt0050976,tt0069467,tt0050986	writer

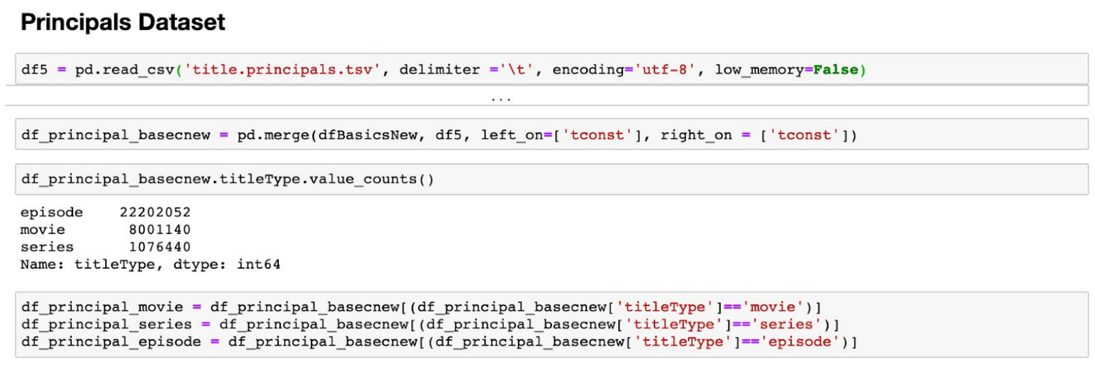
- Using this information, we created 3 entities viz. actor(which contains actor and actress information), writer and director.



```
df4_final_actor_actress = df4_final[(df4_final["primaryProfession"] == "actor") |
                                    (df4_final["primaryProfession"] == "actress")]
df4_final_director = df4_final[df4_final["primaryProfession"] == "director"]
df4_final_writer = df4_final[df4_final["primaryProfession"] == "writer"]
```

Title.principals.tsv

- We first merged this dataset with datasets we created, containing movie, series and episode information.

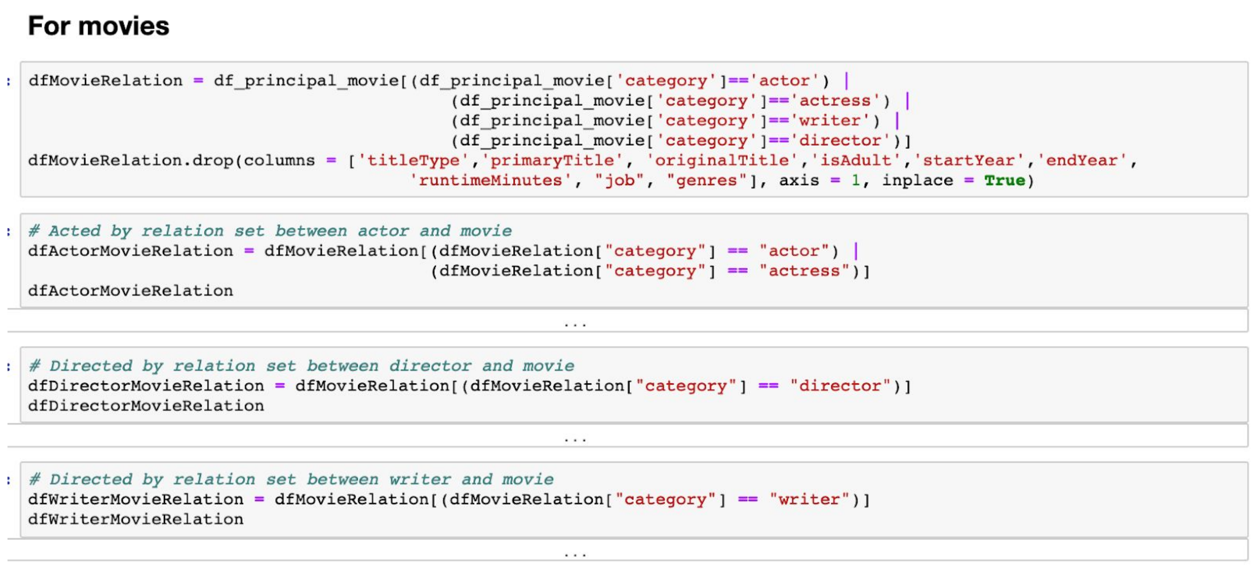


Principals Dataset

```
df5 = pd.read_csv('title.principals.tsv', delimiter ='\t', encoding='utf-8', low_memory=False)
...

df_principal_basecnew = pd.merge(dfBasicsNew, df5, left_on=['tconst'], right_on = ['tconst'])

df_principal_basecnew.titleType.value_counts()

episode    22202052
movie       8001140
series      1076440
Name: titleType, dtype: int64

df_principal_movie = df_principal_basecnew[(df_principal_basecnew['titleType']=='movie')]
df_principal_series = df_principal_basecnew[(df_principal_basecnew['titleType']=='series')]
df_principal_episode = df_principal_basecnew[(df_principal_basecnew['titleType']=='episode')]
```

- Then, we segregated this new dataset according to the title type and got 3 datasets individually containing principal information of movies, series and episode.

- For Movies



For movies

```
dfMovieRelation = df_principal_movie[(df_principal_movie['category']=='actor') |
                                     (df_principal_movie['category']=='actress') |
                                     (df_principal_movie['category']=='writer') |
                                     (df_principal_movie['category']=='director')]
dfMovieRelation.drop(columns = ['titleType','primaryTitle', 'originalTitle','isAdult','startYear','endYear',
                                'runtimeMinutes', "job", "genres"], axis = 1, inplace = True)
```

```
# Acted by relation set between actor and movie
dfActorMovieRelation = dfMovieRelation[(dfMovieRelation["category"] == "actor") |
                                       (dfMovieRelation["category"] == "actress")]
dfActorMovieRelation
```

...

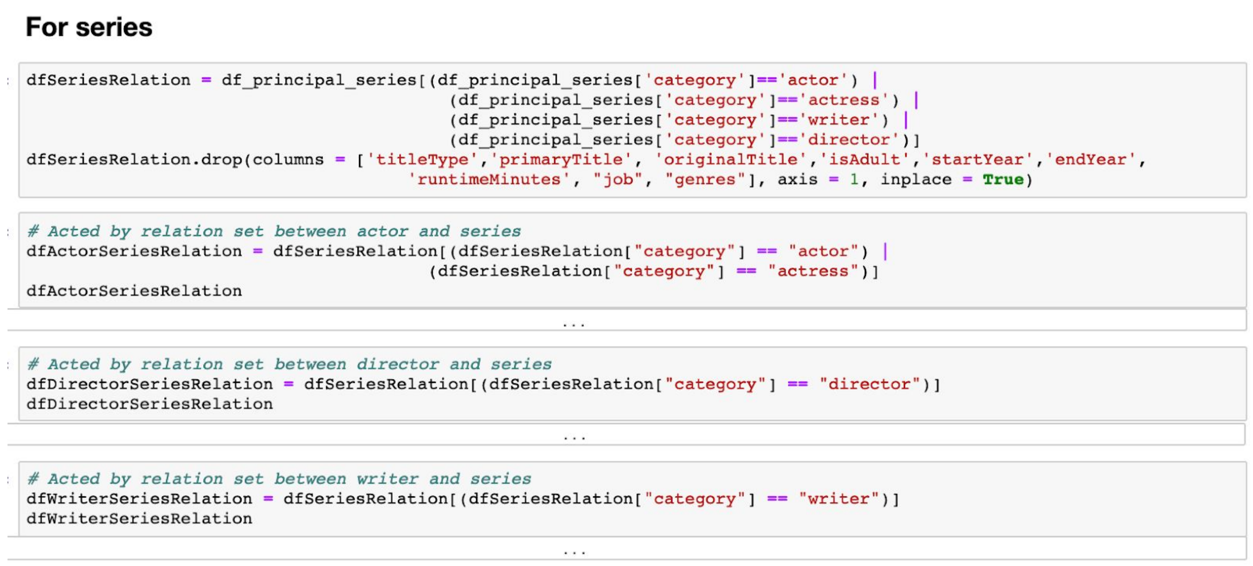
```
# Directed by relation set between director and movie
dfDirectorMovieRelation = dfMovieRelation[(dfMovieRelation["category"] == "director")]
dfDirectorMovieRelation
```

...

```
# Directed by relation set between writer and movie
dfWriterMovieRelation = dfMovieRelation[(dfMovieRelation["category"] == "writer")]
dfWriterMovieRelation
```

...

-
- For Series and by extension and their episodes



For series

```
dfSeriesRelation = df_principal_series[(df_principal_series['category']=='actor') |
                                      (df_principal_series['category']=='actress') |
                                      (df_principal_series['category']=='writer') |
                                      (df_principal_series['category']=='director')]
dfSeriesRelation.drop(columns = ['titleType','primaryTitle', 'originalTitle','isAdult','startYear','endYear',
                                'runtimeMinutes', "job", "genres"], axis = 1, inplace = True)
```

```
# Acted by relation set between actor and series
dfActorSeriesRelation = dfSeriesRelation[(dfSeriesRelation["category"] == "actor") |
                                        (dfSeriesRelation["category"] == "actress")]
dfActorSeriesRelation
```

...

```
# Acted by relation set between director and series
dfDirectorSeriesRelation = dfSeriesRelation[(dfSeriesRelation["category"] == "director")]
dfDirectorSeriesRelation
```

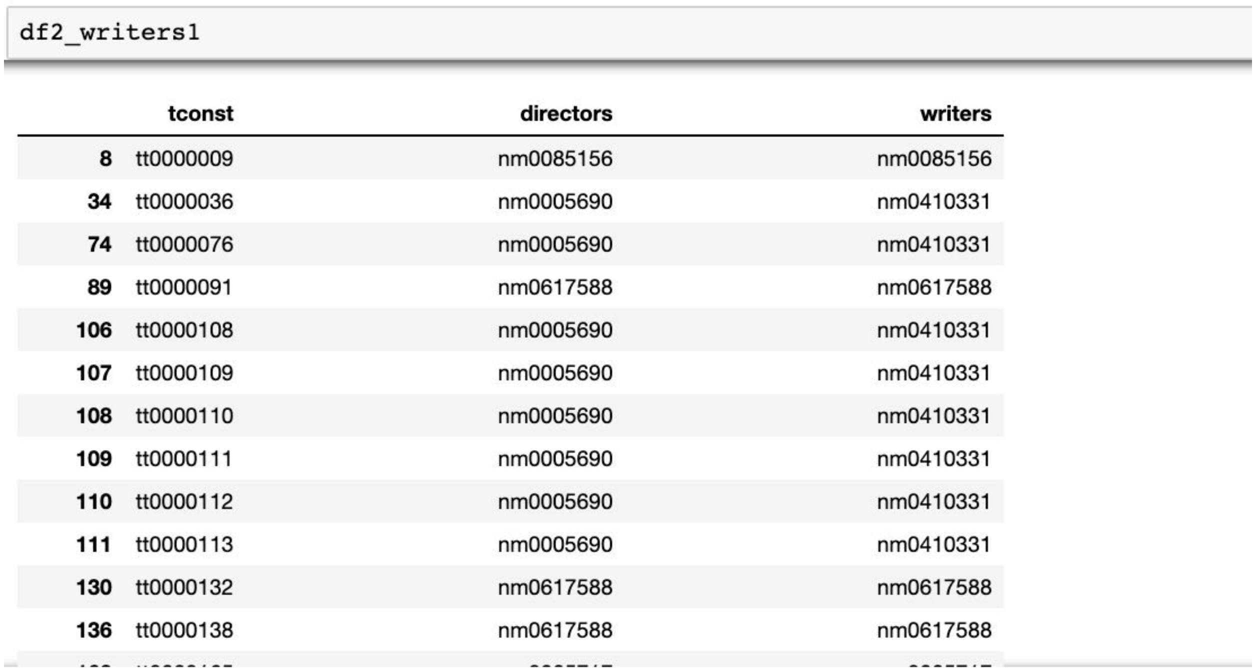
...

```
# Acted by relation set between writer and series
dfWriterSeriesRelation = dfSeriesRelation[(dfSeriesRelation["category"] == "writer")]
dfWriterSeriesRelation
```

...

Title.crew.tsv

- In this dataset, both directors and writers columns contains multi value cells. So we generated number of rows according to their multiple values. We also removed null values.



```
df2_writers1
```

	tconst	directors	writers
8	tt0000009	nm0085156	nm0085156
34	tt0000036	nm0005690	nm0410331
74	tt0000076	nm0005690	nm0410331
89	tt0000091	nm0617588	nm0617588
106	tt0000108	nm0005690	nm0410331
107	tt0000109	nm0005690	nm0410331
108	tt0000110	nm0005690	nm0410331
109	tt0000111	nm0005690	nm0410331
110	tt0000112	nm0005690	nm0410331
111	tt0000113	nm0005690	nm0410331
130	tt0000132	nm0617588	nm0617588
136	tt0000138	nm0617588	nm0617588