In [1]:
```
import pandas as pd
import json
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
file_dir = 'C:/Users/14153/UCDavis_Code/Bootcamp/ETL/Kaggle/'
```

In [3]:
```
kaggle_metadata = pd.read_csv(f'{file_dir}movies_metadata.csv', low_memory=False)
ratings = pd.read_csv(f'{file_dir}ratings.csv')
```

In [4]:
```
kaggle_metadata.head(5)
```

Out[4]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...	re
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt0114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	...	
1	False	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	tt0113497	en	Jumanji	When siblings Judy and Peter discover an encha...	...	
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	NaN	15602	tt0113228	en	Grumpier Old Men	A family wedding reignites the ancient feud be...	...	
3	False	NaN	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	31357	tt0114885	en	Waiting to Exhale	Cheated on, mistreated and stepped on, the wom...	...	
4	False	{'id': 96871, 'name': 'Father of the Bride Col...	0	[{'id': 35, 'name': 'Comedy'}]	NaN	11862	tt0113041	en	Father of the Bride Part II	Just when George Banks has recovered from his ...	...	

5 rows × 24 columns

In [5]:
```
kaggle_metadata.sample(n=5)
```

Out[5]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title
4339	False	NaN	46630000	[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...	NaN	14506	tt0096764	en	The Adventures of Baron Munchausen
18046	False	NaN	0	[{'id': 35, 'name': 'Comedy'}, {'id': 80, 'nam...	NaN	46934	tt1183911	en	Mr. Nice
42065	False	NaN	0	[{'id': 18, 'name': 'Drama'}, {'id': 10770, 'n...	http://www.hallmarkchannel.com/tulips-in-spring	393658	tt5605328	en	Tulips in Spring
41026	False	{'id': 246839, 'name': 'Center Stage Collectio...	0	[{'id': 10402, 'name': 'Music'}, {'id': 10770,...	http://www.mylifetime.com/movies/center-stage-...	402543	tt5176536	en	Center Stage: On Pointe
19573	False	NaN	0	[{'id': 99, 'name': 'Documentary'}]	NaN	196738	tt0293282	en	Inside: 'Dr. Strangelove or: How I Learned to ...

5 rows × 24 columns

In [6]:
```
kaggle_metadata.tail(5)
```

Out[6]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overv
45461	False	NaN	0	[{'id': 18, 'name': 'Drama'}, {'id': 10751, 'n...	http://www.imdb.com/title/tt6209470/	439050	tt6209470	fa	رگ خواب	Rising falling betw a man wor
45462	False	NaN	0	[{'id': 18, 'name': 'Drama'}]	NaN	111109	tt2028550	tl	Siglo ng Pagluluwal	An a struggle finish his v while
45463	False	NaN	0	[{'id': 28, 'name': 'Action'}, {'id': 18, 'nam...	NaN	67758	tt0303758	en	Betrayal	When one of hits goes wr a professio
45464	False	NaN	0	[]	NaN	227506	tt0008536	en	Satana likuyushchiy	In a small t live brothers, o mir
45465	False	NaN	0	[]	NaN	461257	tt6980792	en	Queerama	50 years decriminalisa of homosexu

5 rows × 24 columns

In [7]:
```
ratings.head(5)
```

Out[7]:

	userId	movieId	rating	timestamp
0	1	110	1.0	1425941529
1	1	147	4.5	1425942435
2	1	858	5.0	1425941523
3	1	1221	5.0	1425941546
4	1	1246	5.0	1425941556

In [8]:
```
ratings.sample(5)
```

Out[8]:

	userId	movieId	rating	timestamp
25556630	265982	3948	3.0	1455938012
18002735	186927	115617	3.5	1443053836
20281382	210792	3839	3.5	1467065569
23303138	241840	4995	3.0	1357167918
7941312	81814	1348	4.0	1447928816

In [9]:
```
ratings.tail(5)
```

Out[9]:

	userId	movieId	rating	timestamp
26024284	270896	58559	5.0	1257031564
26024285	270896	60069	5.0	1257032032
26024286	270896	63082	4.5	1257031764
26024287	270896	64957	4.5	1257033990
26024288	270896	71878	2.0	1257031858

8.3.1 Data Cleaning Strategies

In [10]:
```
with open(f'{file_dir}wikipedia_movies.json', mode='r') as file:
    wiki_movies_raw = json.load(file)
```

In [11]:
```
len(wiki_movies_raw)
```

Out[11]:
```
7311
```

In [12]:
```
wiki_movies_raw[:5]
```

Out[12]:
```
[{'url': 'https://en.wikipedia.org/wiki/The_Adventures_of_Ford_Fairlane',
  'year': 1990,
  'imdb_link': 'https://www.imdb.com/title/tt0098987/',
  'title': 'The Adventures of Ford Fairlane',
  'Directed by': 'Renny Harlin',
  'Produced by': ['Steve Perry', 'Joel Silver'],
  'Screenplay by': ['David Arnott', 'James Cappe', 'Daniel Waters'],
  'Story by': ['David Arnott', 'James Cappe'],
  'Based on': ['Characters', 'by Rex Weiner'],
  'Starring': ['Andrew Dice Clay',
   'Wayne Newton',
   'Priscilla Presley',
   'Lauren Holly',
   'Morris Day',
   'Robert Englund',
   "Ed O'Neill"],
  'Narrated by': 'Andrew "Dice" Clay',
  'Music by': ['Cliff Eidelman', 'Yello'],
  'Cinematography': 'Oliver Wood',
  'Edited by': 'Michael Tronick'
```

In [13]:
```
wiki_movies_raw[-5:]
```

Out[13]:
```
[{'url': 'https://en.wikipedia.org/wiki/Holmes_%26_Watson',
  'year': 2018,
  'imdb_link': 'https://www.imdb.com/title/tt1255919/',
  'title': 'Holmes & Watson',
  'Directed by': 'Etan Cohen',
  'Produced by': ['Will Ferrell',
   'Adam McKay',
   'Jimmy Miller',
   'Clayton Townsend'],
  'Screenplay by': 'Etan Cohen',
  'Based on': ['Sherlock Holmes',
   'and',
   'Dr. Watson',
   'by',
   'Sir Arthur Conan Doyle'],
  'Starring': ['Will Ferrell',
   'John C. Reilly',
   'Rebecca Hall',
   'Rob Brydon',
```

In [14]:
```
wiki_movies_raw[3600:3605]
```

```
  'Screenplay by': 'Sebastian Gutierrez',
  'Based on': ['The Big Bounce', 'by', 'Elmore Leonard'],
  'Starring': ['Owen Wilson',
   'Charlie Sheen',
   'Morgan Freeman',
   'Sara Foster',
   'Gary Sinise'],
  'Music by': 'George S. Clinton',
  'Cinematography': 'Jeffrey L. Kimball',
  'Edited by': 'Barry Malkin',
  'Productioncompany ': 'Shangri-La Entertainment',
  'Distributed by': 'Warner Bros. Pictures',
  'Release date': ['January 30, 2004', '(', '2004-01-30', ')'],
  'Running time': '88 minutes',
  'Language': 'English',
  'Budget': '$50 million',
  'Box office': '$6.8 million'},
 {'url': 'https://en.wikipedia.org/wiki/Birth_(film)',
  'year': 2004,
  'imdb_link': 'https://www.imdb.com/title/tt0337876/',
```

In [15]:
```
wiki_movies_df = pd.DataFrame(wiki_movies_raw)
wiki_movies_df.head(5)
```

Out[15]:

	url	year	imdb_link	title	Directed by	Produced by	Screenplay by	Story by	Ba
0	https://en.wikipedia.org/wiki/The_Adventures_o...	1990.0	https://www.imdb.com/title/tt0098987/	The Adventures of Ford Fairlane	Renny Harlin	[Steve Perry, Joel Silver]	[David Arnott, James Cappe, Daniel Waters]	[David Arnott, James Cappe]	[Cha
1	https://en.wikipedia.org/wiki/After_Dark,_My_S...	1990.0	https://www.imdb.com/title/tt0098994/	After Dark, My Sweet	James Foley	[Ric Kidney, Robert Redlin]	[James Foley, Robert Redlin]	NaN	[th Aft My
2	https://en.wikipedia.org/wiki/Air_America_(film)	1990.0	https://www.imdb.com/title/tt0099005/	Air America	Roger Spottiswoode	Daniel Melnick	[John Eskow, Richard Rush]	NaN	Ame Chr F
3	https://en.wikipedia.org/wiki/Alice_(1990_film)	1990.0	https://www.imdb.com/title/tt0099012/	Alice	Woody Allen	Robert Greenhut	NaN	NaN	
4	https://en.wikipedia.org/wiki/Almost_an_Angel	1990.0	https://www.imdb.com/title/tt0099018/	Almost an Angel	John Cornell	John Cornell	NaN	NaN	

5 rows × 193 columns

In [16]:
```
columns_list = wiki_movies_df.columns.tolist()
columns_list
```

```
 'Audio format',
 'Voices of',
 'Followed by',
 'Composer(s)',
 'Created by',
 'Also known as',
 'Opening theme',

 'No. of episodes',
 'Preceded by',
 'Author',
 'Publisher',
 'Publication date',
 'Media type',
 'Pages',
 'ISBN',
 'OCLC',
 'LC Class',
 'Cover artist',
 'Series',
```

In [17]:
```
###List Comprehension to Filter to only the movies wtih a director and IMBD link
wiki_movies = [movie for movie in wiki_movies_raw
               if ('Director' in movie or 'Directed by' in movie)
                   and 'imdb_link' in movie]
len(wiki_movies)
```

Out[17]: 7080

8.3.5 Create a Function to Clean Data

In [18]:
```
wiki_movies = [movie for movie in wiki_movies_raw
               if ('Director' in movie or 'Directed by' in movie)
                   and 'imdb_link' in movie
                   and 'No. of episodes' not in movie]
```

In [19]:
```
def clean_movie(movie):
    movie = dict(movie) #create a non-destructive copy
    return movie
```

In [20]:
```
wiki_movies_df[wiki_movies_df['Arabic'].notnull()]
```

Out[20]:

	url	year	imdb_link	title	Directed by	Produced by	Screenplay by	Story by	E
7060	https://en.wikipedia.org/wiki/The_Insult_(film)	2018.0	https://www.imdb.com/title/tt7048622/	The Insult	Ziad Doueiri	[Rachid Bouchareb, Jean Bréhat, Julie Gayet, A...	NaN	NaN	
7293	https://en.wikipedia.org/wiki/Capernaum_(film)	2018.0	https://www.imdb.com/title/tt8267604/	Capernaum	Nadine Labaki	[Michel Merkt, Khaled Mouzanar]	[Nadine Labaki, Jihad Hojaily, Michelle Keserw...	[Georges Khabbaz, Nadine Labaki, Michelle Kese...	

2 rows × 193 columns

In [21]:
```
wiki_movies_df[wiki_movies_df['Arabic'].notnull()]['url']
```

Out[21]:
```
7060    https://en.wikipedia.org/wiki/The_Insult_(film) (https://en.wikipedia.org/wiki/The_Insult_(film))
7293     https://en.wikipedia.org/wiki/Capernaum_(film) (https://en.wikipedia.org/wiki/Capernaum_(film))
Name: url, dtype: object
```

In [22]:
```
sorted(wiki_movies_df.columns.tolist())
```

```
 'Founders',
 'French',
 'Full name',
 'Gender',
 'Genre',
 'Genre(s)',
 'Genres',
 'Gwoyeu Romatzyh',
 'Hangul',
 'Hanyu Pinyin',
 'Headquarters',
 'Hebrew',
 'Height',
 'Hepburn',
 'Hokkien POJ',
 'IPA',
 'ISBN',
 'Illustrator',
 'Industry',
 'Instruments',
```

In [23]:
```
def clean_movie(movie):
    movie = dict(movie) #create a non-destructive copy
    alt_titles = {}
    return movie
```

In [24]:
```
def clean_movie(movie):
    movie = dict(movie) #create a non-destructive copy
    alt_titles = {}
    for key in ['Also known as','Arabic','Cantonese','Chinese','French',
                'Hangul','Hebrew','Hepburn','Japanese','Literally',
                'Mandarin','McCune-Reischauer','Original title','Polish',
                'Revised Romanization','Romanized','Russian',
                'Simplified','Traditional','Yiddish']:

            return movie
```

In [25]:
```
def clean_movie(movie):
    movie = dict(movie) #create a non-destructive copy
    alt_titles = {}
    for key in ['Also known as','Arabic','Cantonese','Chinese','French',
                'Hangul','Hebrew','Hepburn','Japanese','Literally',
                'Mandarin','McCune-Reischauer','Original title','Polish',
                'Revised Romanization','Romanized','Russian',
                'Simplified','Traditional','Yiddish']:
        if key in movie:

            return movie
```

In [26]:
```
def clean_movie(movie):
    movie = dict(movie) #create a non-destructive copy
    alt_titles = {}
    for key in ['Also known as','Arabic','Cantonese','Chinese','French',
                'Hangul','Hebrew','Hepburn','Japanese','Literally',
                'Mandarin','McCune-Reischauer','Original title','Polish',
                'Revised Romanization','Romanized','Russian',
                'Simplified','Traditional','Yiddish']:
        if key in movie:
            alt_titles[key] = movie[key]
            movie.pop(key)


    return movie
```

In [27]:
```
def clean_movie(movie):
    movie = dict(movie) #create a non-destructive copy
    alt_titles = {}
    for key in ['Also known as','Arabic','Cantonese','Chinese','French',
                'Hangul','Hebrew','Hepburn','Japanese','Literally',
                'Mandarin','McCune-Reischauer','Original title','Polish',
                'Revised Romanization','Romanized','Russian',
                'Simplified','Traditional','Yiddish']:
        if key in movie:
            alt_titles[key] = movie[key]
            movie.pop(key)
    if len(alt_titles) > 0:
        movie['alt_titles'] = alt_titles

    return movie
```

In [28]:
```
clean_movies = [clean_movie(movie) for movie in wiki_movies]
```

In [29]:
```
wiki_movies_df = pd.DataFrame(clean_movies)
sorted(wiki_movies_df.columns.tolist())
```

Out[29]:
```
['Adaptation by',
 'Animation by',
 'Audio format',
 'Based on',
 'Box office',
 'Budget',
 'Cinematography',
 'Color process',
 'Composer(s)',
 'Country',
 'Country of origin',
 'Created by',
 'Directed by',
 'Director',
 'Distributed by',
 'Distributor',
 'Edited by',
 'Editor(s)',
 'Executive producer(s)',
 'Followed by',
 'Genre',
 'Label',
 'Language',
 'Length',
 'Music by',
 'Narrated by',
 'Original language(s)',
 'Original network',
 'Original release',
 'Picture format',
 'Preceded by',
 'Produced by',
 'Producer',
 'Producer(s)',
 'Production company(s)',
 'Production location(s)',
 'Productioncompanies ',
 'Productioncompany ',
 'Recorded',
 'Release date',
 'Released',
 'Running time',
 'Screen story by',
 'Screenplay by',
```

```
 'Starring',
 'Story by',
 'Suggested by',
 'Theme music composer',
 'Venue',
 'Voices of',
 'Written by',
 'alt_titles',
 'imdb_link',
 'title',
 'url',
 'year']
```

In [34]:

```
    def change_column_name(old_name, new_name):
        if old_name in movie:
            movie[new_name] = movie.pop(old_name)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_26064/291284349.py in <module>
      2     if old_name in movie:
      3         movie[new_name] = movie.pop(old_name)
----> 4 change_column_name('Directed by', 'Director')
      5 change_column_name

~\AppData\Local\Temp/ipykernel_26064/291284349.py in change_column_name(old_name, new_name)
      1 def change_column_name(old_name, new_name):
----> 2     if old_name in movie:
      3         movie[new_name] = movie.pop(old_name)
      4 change_column_name('Directed by', 'Director')
      5 change_column_name

NameError: name 'movie' is not defined
```

In [31]:
```
change_column_name('Directed by', 'Director')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_26064/1532660836.py in <module>
----> 1 change_column_name('Directed by', 'Director')

~\AppData\Local\Temp/ipykernel_26064/1020343741.py in change_column_name(old_name, new_name)
      1 def change_column_name(old_name, new_name):
----> 2     if old_name in movie:
      3         movie[new_name] = movie.pop(old_name)

NameError: name 'movie' is not defined
```

In [ ]:
```
change_column_name('Adaptation by', 'Writer(s)')
```

In [32]:
```
# Merge column names
 def change_column_name(old_name, new_name):
     if old_name in movie:
         movie[new_name] = movie.pop(old_name)
change_column_name('Adaptation by', 'Writer(s)')
change_column_name
```

```
  File "C:\Users\14153\AppData\Local\Temp/ipykernel_26064/242869450.py", line 2
    def change_column_name(old_name, new_name):
    ^
IndentationError: unexpected indent
```

In [ ]:

In [ ]: