In [210]:
```
import pandas as pd
import numpy as np
pd.options.display.max_columns = 30
```

In [211]:
```
df = pd.read_csv("X:\Data Science\Projects\Work with Pandas, SQL Databases
```

In [212]:
```
df.head()
```

Out[212]:

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt01147
1	False	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	tt01134
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	NaN	15602	tt01132
3	False	NaN	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	31357	tt01148
4	False	{'id': 96871, 'name': 'Father of the Bride Col...	0	[{'id': 35, 'name': 'Comedy'}]	NaN	11862	tt01130

In [213]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 24 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   adult                  45466 non-null  object 
 1   belongs_to_collection  4494 non-null   object 
 2   budget                 45466 non-null  object 
 3   genres                 45466 non-null  object 
 4   homepage               7782 non-null   object 
 5   id                     45466 non-null  object 
 6   imdb_id                45449 non-null  object 
 7   original_language      45455 non-null  object 
 8   original_title         45466 non-null  object 
 9   overview               44512 non-null  object 
 10  popularity             45461 non-null  object 
 11  poster_path            45080 non-null  object 
 12  production_companies   45463 non-null  object 
 13  production_countries   45463 non-null  object 
 14  release_date           45379 non-null  object 
 15  revenue                45460 non-null  float64
 16  runtime                45203 non-null  float64
 17  spoken_languages       45460 non-null  object 
 18  status                 45379 non-null  object 
 19  tagline                20412 non-null  object 
 20  title                  45460 non-null  object 
 21  video                  45460 non-null  object 
 22  vote_average           45460 non-null  float64
 23  vote_count             45460 non-null  float64
dtypes: float64(4), object(20)
memory usage: 8.3+ MB
```

In [214]:
```
df.genres[0]
```

Out[214]:
```
"[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id': 1075
1, 'name': 'Family'}]"
```

In [215]:
```
df.belongs_to_collection[0]
```

Out[215]:
```
"{'id': 10194, 'name': 'Toy Story Collection', 'poster_path': '/7G9915LfUQ2lV
fwMEEhDsn3kT4B.jpg', 'backdrop_path': '/9FBwqcd9IRruEDUrTdcaafOMKUq.jpg'}"
```

Dropping irrelevent Columns

In [216]:
```
df.adult.value_counts()
```

Out[216]:
```
False
45454
True
9
 - Written by Ørnås
1
 Rune Balot goes to a casino connected to the October corporation to try to w
rap up her case once and for all.                              1
 Avalanche Sharks tells the story of a bikini contest that turns into a horri
fying affair when it is hit by a shark avalanche.              1
Name: adult, dtype: int64
```

In [217]:
```
df.drop(columns = ["adult"], inplace = True)
```

In [218]:
```
df.drop(columns = ["imdb_id"], inplace = True)
```

In [219]:
```
df.drop(columns = ["original_title"], inplace = True)
```

In [220]:
```
df.drop(columns = ["video"], inplace = True)
```

In [221]:
```
df.drop(columns = ["homepage"], inplace = True)
```

How to handle stringified JSON columns (Part 1)

In [222]:
```
import json
import ast
```

In [223]:
```
json_col = ["belongs_to_collection", "genres", "production_countries","pro
```

In [224]:
```
df["belongs_to_collection"][0]
```

Out[224]:
```
"{'id': 10194, 'name': 'Toy Story Collection', 'poster_path': '/7G9915LfUQ2lV
fwMEEhDsn3kT4B.jpg', 'backdrop_path': '/9FBwqcd9IRruEDUrTdcaafOMKUq.jpg'}"
```

In [225]:
```
json1 = "{'dog':3,'cat':5}"
```

In [6]:
```
json.loads(json1)
```

In [227]:
```
json2 = '{"dog":3,"cat":5}'
```

In [228]:
```
json.loads(json2)
```

Out[228]:
```
{'dog': 3, 'cat': 5}
```

In [229]:
```
json1.replace("'",'"')
```

Out[229]:
```
'{"dog":3,"cat":5}'
```

In [230]:
```
json.loads(json1.replace("'",'"'))
```

Out[230]:
```
{'dog': 3, 'cat': 5}
```

In [231]:
```
df.genres.apply(lambda x: json.loads(x.replace("'",'"')))[0]
```

Out[231]:
```
[{'id': 16, 'name': 'Animation'},
 {'id': 35, 'name': 'Comedy'},
 {'id': 10751, 'name': 'Family'}]
```

In [232]:
```
ast.literal_eval(json1)
```

Out[232]:
```
{'dog': 3, 'cat': 5}
```

In [233]:
```
ast.literal_eval(json2)
```

Out[233]:
```
{'dog': 3, 'cat': 5}
```

In [234]:
```
df.genres.apply(ast.literal_eval)[0]
```

Out[234]:
```
[{'id': 16, 'name': 'Animation'},
 {'id': 35, 'name': 'Comedy'},
 {'id': 10751, 'name': 'Family'}]
```

In [235]:
```
df.genres = df.genres.apply(ast.literal_eval)
```

In [7]:
```
df.loc[:,json_col].apply(ast.literal_eval, axis = 0)
```

In [ ]:
```
ast.literal_eval(0)
```

How to handle stringfied JSON columns

In [237]:
```
import numpy as np
```

In [238]:
```
df.belongs_to_collection
```

Out[238]:
```
0        {'id': 10194, 'name': 'Toy Story Collection', ...
1                                                      NaN
2        {'id': 119050, 'name': 'Grumpy Old Men Collect...
3                                                      NaN
4        {'id': 96871, 'name': 'Father of the Bride Col...
                               ...
45461                                                  NaN
45462                                                  NaN
45463                                                  NaN
45464                                                  NaN
45465                                                  NaN
Name: belongs_to_collection, Length: 45466, dtype: object
```

In [239]:
```
df.belongs_to_collection.apply(lambda x: isinstance(x, str))
```

Out[239]:
```
0         True
1        False
2         True
3        False
4         True
         ...
45461    False
45462    False
45463    False
45464    False
45465    False
Name: belongs_to_collection, Length: 45466, dtype: bool
```

In [240]:
```
df.belongs_to_collection = df.belongs_to_collection.apply(lambda x: ast.li
```

In [241]:
```
df.belongs_to_collection[0]
```

Out[241]:
```
{'id': 10194,
 'name': 'Toy Story Collection',
 'poster_path': '/7G9915LfUQ2lVfwMEEhDsn3kT4B.jpg',
 'backdrop_path': '/9FBwqcd9IRruEDUrTdcaafOMKUq.jpg'}
```

In [242]:
```
df.spoken_languages
```

Out[242]:
```
0                 [{'iso_639_1': 'en', 'name': 'English'}]
1        [{'iso_639_1': 'en', 'name': 'English'}, {'iso...
2                 [{'iso_639_1': 'en', 'name': 'English'}]
3                 [{'iso_639_1': 'en', 'name': 'English'}]
4                 [{'iso_639_1': 'en', 'name': 'English'}]
                               ...
45461               [{'iso_639_1': 'fa', 'name': 'فارسی'}]
45462                    [{'iso_639_1': 'tl', 'name': ''}]
45463             [{'iso_639_1': 'en', 'name': 'English'}]
45464                                                   []
45465             [{'iso_639_1': 'en', 'name': 'English'}]
Name: spoken_languages, Length: 45466, dtype: object
```

In [243]:
```
df.spoken_languages = df.spoken_languages.apply(lambda x: ast.literal_eval
```

In [244]:
```
df.production_countries
```

Out[244]:
```
0        [{'iso_3166_1': 'US', 'name': 'United States o...
1        [{'iso_3166_1': 'US', 'name': 'United States o...
2        [{'iso_3166_1': 'US', 'name': 'United States o...
3        [{'iso_3166_1': 'US', 'name': 'United States o...
4        [{'iso_3166_1': 'US', 'name': 'United States o...
                               ...                        
45461               [{'iso_3166_1': 'IR', 'name': 'Iran'}]
45462        [{'iso_3166_1': 'PH', 'name': 'Philippines'}]
45463    [{'iso_3166_1': 'US', 'name': 'United States o...
45464             [{'iso_3166_1': 'RU', 'name': 'Russia'}]
45465     [{'iso_3166_1': 'GB', 'name': 'United Kingdom'}]
Name: production_countries, Length: 45466, dtype: object
```

In [245]:
```
df.production_countries = df.production_countries.apply(lambda x: ast.lite
```

In [246]:
```
df.production_companies
```

Out[246]:
```
0           [{'name': 'Pixar Animation Studios', 'id': 3}]
1        [{'name': 'TriStar Pictures', 'id': 559}, {'na...
2        [{'name': 'Warner Bros.', 'id': 6194}, {'name'...
3        [{'name': 'Twentieth Century Fox Film Corporat...
4        [{'name': 'Sandollar Productions', 'id': 5842}...
                               ...                        
45461                                                   []
45462               [{'name': 'Sine Olivia', 'id': 19653}]
45463    [{'name': 'American World Pictures', 'id': 6165}]
45464                 [{'name': 'Yermoliev', 'id': 88753}]
45465                                                   []
Name: production_companies, Length: 45466, dtype: object
```

In [247]:
```
df.production_companies = df.production_companies.apply(lambda x: ast.lite
```

In [248]:
```
df.production_companies
```

Out[248]:
```
0           [{'name': 'Pixar Animation Studios', 'id': 3}]
1        [{'name': 'TriStar Pictures', 'id': 559}, {'na...
2        [{'name': 'Warner Bros.', 'id': 6194}, {'name'...
3        [{'name': 'Twentieth Century Fox Film Corporat...
4        [{'name': 'Sandollar Productions', 'id': 5842}...
                               ...                        
45461                                                   []
45462               [{'name': 'Sine Olivia', 'id': 19653}]
45463    [{'name': 'American World Pictures', 'id': 6165}]
45464                 [{'name': 'Yermoliev', 'id': 88753}]
45465                                                   []
Name: production_companies, Length: 45466, dtype: object
```

How to flatten nested Columns

In [249]:
```
df.belongs_to_collection[0]
```

Out[249]:
```
{'id': 10194,
 'name': 'Toy Story Collection',
 'poster_path': '/7G9915LfUQ2lVfwMEEhDsn3kT4B.jpg',
 'backdrop_path': '/9FBwqcd9IRruEDUrTdcaafOMKUq.jpg'}
```

In [250]:
```
df.belongs_to_collection = df.belongs_to_collection.apply(lambda x: x['nam
```

In [251]:
```
df.belongs_to_collection.value_counts(dropna= False).head(20)
```

Out[251]:
```
NaN                                      40975
The Bowery Boys                             29
Totò Collection                             27
James Bond Collection                       26
Zatôichi: The Blind Swordsman               26
The Carry On Collection                     25
Pokémon Collection                          22
Charlie Chan (Sidney Toler) Collection      21
Godzilla (Showa) Collection                 16
Uuno Turhapuro                              15
Dragon Ball Z (Movie) Collection            15
Charlie Chan (Warner Oland) Collection      15
The Land Before Time Collection             14
Monster High Collection                     14
Sharpe Collection                           13
George Carlin Comedy Collection             13
Johan Falk GSI Collection                   12
Sherlock Holmes (1939 series)               12
Friday the 13th Collection                  12
The Amityville Horror Collection            12
Name: belongs_to_collection, dtype: int64
```

In [252]:
```
df.genres[0]
```

Out[252]:
```
[{'id': 16, 'name': 'Animation'},
 {'id': 35, 'name': 'Comedy'},
 {'id': 10751, 'name': 'Family'}]
```

In [253]:
```
df.genres = df.genres.apply(lambda x: "|".join(i["name"] for i in x))
```

In [254]:
```
df.genres
```

Out[254]:
```
0          Animation|Comedy|Family
1         Adventure|Fantasy|Family
2                   Romance|Comedy
3             Comedy|Drama|Romance
4                           Comedy
                   ...
45461                 Drama|Family
45462                        Drama
45463        Action|Drama|Thriller
45464
45465
Name: genres, Length: 45466, dtype: object
```

In [255]:
```
df.genres.value_counts(dropna = False).head(20)
```

Out[255]:
```
Drama                   5000
Comedy                  3621
Documentary             2723
                        2442
Drama|Romance           1301
Comedy|Drama            1135
Horror                   974
Comedy|Romance           930
Comedy|Drama|Romance     593
Drama|Comedy             532
Horror|Thriller          528
Drama|Thriller           497
Thriller                 465
Crime|Drama              430
Romance|Drama            343
Western                  318
Action|Thriller          301
Drama|Foreign            283
Action                   278
Drama|History            267
Name: genres, dtype: int64
```

In [256]:
```
df.genres.replace("", np.nan, inplace = True)
```

In [257]:
```
df.spoken_languages
```

Out[257]:
```
0                  [{'iso_639_1': 'en', 'name': 'English'}]
1         [{'iso_639_1': 'en', 'name': 'English'}, {'iso...
2                  [{'iso_639_1': 'en', 'name': 'English'}]
3                  [{'iso_639_1': 'en', 'name': 'English'}]
4                  [{'iso_639_1': 'en', 'name': 'English'}]
                                ...
45461                [{'iso_639_1': 'fa', 'name': 'فارسی'}]
45462                     [{'iso_639_1': 'tl', 'name': ''}]
45463              [{'iso_639_1': 'en', 'name': 'English'}]
45464                                                    []
45465              [{'iso_639_1': 'en', 'name': 'English'}]
Name: spoken_languages, Length: 45466, dtype: object
```

In [258]:
```
df.spoken_languages = df.spoken_languages.apply(lambda x: "|".join(i["name
```

In [259]:
```
df.spoken_languages.value_counts(dropna = False).head(20)
```

Out[259]:
```
English              22395
                      3952
Français              1853
日本語                  1289
Italiano              1218
Español                902
Русский                807
Deutsch                762
English|Français       681
English|Español        572
हिन्दी                  481
English|Deutsch        462
한국어/조선말               425
普通话                   347
English|Italiano       326
svenska                311
No Language            303
suomi                  275
Português              275
Polski                 213
Name: spoken_languages, dtype: int64
```

In [260]:
```
df.spoken_languages.replace("", np.nan, inplace = True)
```

In [261]:
```
df.production_countries
```

Out[261]:
```
0        [{'iso_3166_1': 'US', 'name': 'United States o...
1        [{'iso_3166_1': 'US', 'name': 'United States o...
2        [{'iso_3166_1': 'US', 'name': 'United States o...
3        [{'iso_3166_1': 'US', 'name': 'United States o...
4        [{'iso_3166_1': 'US', 'name': 'United States o...
                               ...
45461               [{'iso_3166_1': 'IR', 'name': 'Iran'}]
45462        [{'iso_3166_1': 'PH', 'name': 'Philippines'}]
45463    [{'iso_3166_1': 'US', 'name': 'United States o...
45464             [{'iso_3166_1': 'RU', 'name': 'Russia'}]
45465     [{'iso_3166_1': 'GB', 'name': 'United Kingdom'}]
Name: production_countries, Length: 45466, dtype: object
```

In [262]:
```
df.production_countries = df.production_countries.apply(lambda x: "|".join
```

In [263]:
```
df.production_countries
```

Out[263]:
```
0        United States of America
1        United States of America
2        United States of America
3        United States of America
4        United States of America
                   ...
45461                        Iran
45462                 Philippines
45463    United States of America
45464                      Russia
45465              United Kingdom
Name: production_countries, Length: 45466, dtype: object
```

In [264]:
```
df.production_countries.value_counts(dropna = False).head(20)
```

Out[264]:
```
United States of America                  17851
                                           6282
United Kingdom                             2238
France                                     1654
Japan                                      1356
Italy                                      1030
Canada                                      840
Germany                                     749
India                                       735
Russia                                      735
United Kingdom|United States of America     569
South Korea                                 432
Spain                                       398
Hong Kong                                   365
Canada|United States of America             365
Australia                                   336
Sweden                                      332
Finland                                     271
France|Italy                                235
Germany|United States of America            214
Name: production_countries, dtype: int64
```

In [265]:
```
df.production_countries.replace("", np.nan, inplace = True)
```

In [266]:
```
df.production_companies
```

Out[266]:
```
0            [{'name': 'Pixar Animation Studios', 'id': 3}]
1         [{'name': 'TriStar Pictures', 'id': 559}, {'na...
2         [{'name': 'Warner Bros.', 'id': 6194}, {'name'...
3         [{'name': 'Twentieth Century Fox Film Corporat...
4         [{'name': 'Sandollar Productions', 'id': 5842}...
                                ...
45461                                                    []
45462                 [{'name': 'Sine Olivia', 'id': 19653}]
45463     [{'name': 'American World Pictures', 'id': 6165}]
45464                   [{'name': 'Yermoliev', 'id': 88753}]
45465                                                    []
Name: production_companies, Length: 45466, dtype: object
```

In [267]:
```
df.production_companies = df.production_companies.apply(lambda x: "|".join
```

In [268]:
```
df.production_companies
```

Out[268]:
```
0                                     Pixar Animation Studios
1         TriStar Pictures|Teitler Film|Interscope Commu...
2                                 Warner Bros.|Lancaster Gate
3                       Twentieth Century Fox Film Corporation
4                  Sandollar Productions|Touchstone Pictures
                                   ...                        
45461                                                         
45462                                             Sine Olivia
45463                                  American World Pictures
45464                                               Yermoliev
45465                                                         
Name: production_companies, Length: 45466, dtype: object
```

In [269]:
```
df.production_companies.value_counts(dropna = False).head(20)
```

Out[269]:
```
                                          11875
Metro-Goldwyn-Mayer (MGM)                   742
Warner Bros.                                540
Paramount Pictures                          505
Twentieth Century Fox Film Corporation      439
Universal Pictures                          320
RKO Radio Pictures                          247
Columbia Pictures Corporation               207
Columbia Pictures                           146
Mosfilm                                     145
Walt Disney Pictures                         85
Universal International Pictures (UI)        82
New Line Cinema                              75
Walt Disney Productions                      75
Shaw Brothers                                71
Touchstone Pictures                          70
Toho Company                                 65
TriStar Pictures                             62
Orion Pictures                               61
Hammer Film Productions                      60
Name: production_companies, dtype: int64
```

In [270]:
```
df.production_companies.replace("", np.nan, inplace = True)
```

In [271]:
```
df.isna().sum()
```

Out[271]:
```
belongs_to_collection    40975
budget                       0
genres                    2442
id                           0
original_language           11
overview                   954
popularity                   5
poster_path                386
production_companies     11881
production_countries      6288
release_date                87
revenue                      6
runtime                    263
spoken_languages          3958
status                      87
tagline                  25054
title                        6
vote_average                 6
vote_count                   6
dtype: int64
```

In [272]:
```
pd.read_csv("X:\Data Science\Projects\Work with Pandas, SQL Databases, JSO
```

Out[272]:
```
adult                        0
belongs_to_collection    40972
budget                       0
genres                       0
homepage                 37684
id                           0
imdb_id                     17
original_language           11
original_title               0
overview                   954
popularity                   5
poster_path                386
production_companies         3
production_countries         3
release_date                87
revenue                      6
runtime                    263
spoken_languages             6
status                      87
tagline                  25054
title                        6
video                        6
vote_average                 6
vote_count                   6
dtype: int64
```

Cleaning Numerical Columns(Part 1)

In [273]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   belongs_to_collection  4491 non-null   object 
 1   budget                 45466 non-null  object 
 2   genres                 43024 non-null  object 
 3   id                     45466 non-null  object 
 4   original_language      45455 non-null  object 
 5   overview               44512 non-null  object 
 6   popularity             45461 non-null  object 
 7   poster_path            45080 non-null  object 
 8   production_companies   33585 non-null  object 
 9   production_countries   39178 non-null  object 
 10  release_date           45379 non-null  object 
 11  revenue                45460 non-null  float64
 12  runtime                45203 non-null  float64
 13  spoken_languages       41508 non-null  object 
 14  status                 45379 non-null  object 
 15  tagline                20412 non-null  object 
 16  title                  45460 non-null  object 
 17  vote_average           45460 non-null  float64
 18  vote_count             45460 non-null  float64
dtypes: float64(4), object(15)
memory usage: 6.6+ MB
```

In [3]:
```
df.budget.astype("float")
```

In [4]:
```
pd.to_numeric(df.budget)
```

In [276]:
```
df.budget = pd.to_numeric(df.budget, errors = "coerce")
```

In [277]:
```
df.budget.value_counts(dropna = False)
```

Out[277]:
```
0.0           36573
5000000.0       286
10000000.0      259
20000000.0      243
2000000.0       242
              ...  
9750000.0         1
7275000.0         1
78146652.0        1
280.0             1
1254040.0         1
Name: budget, Length: 1224, dtype: int64
```

In [278]:
```
df.budget = df.budget.replace(0, np.nan)
```

In [279]:
```
df.budget = df.budget.div(1000000)
```

In [280]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   belongs_to_collection  4491 non-null   object 
 1   budget                 8890 non-null   float64
 2   genres                 43024 non-null  object 
 3   id                     45466 non-null  object 
 4   original_language      45455 non-null  object 
 5   overview               44512 non-null  object 
 6   popularity             45461 non-null  object 
 7   poster_path            45080 non-null  object 
 8   production_companies   33585 non-null  object 
 9   production_countries   39178 non-null  object 
 10  release_date           45379 non-null  object 
 11  revenue                45460 non-null  float64
 12  runtime                45203 non-null  float64
 13  spoken_languages       41508 non-null  object 
 14  status                 45379 non-null  object 
 15  tagline                20412 non-null  object 
 16  title                  45460 non-null  object 
 17  vote_average           45460 non-null  float64
 18  vote_count             45460 non-null  float64
dtypes: float64(5), object(14)
memory usage: 6.6+ MB
```

In [281]:
```
df.revenue.value_counts(dropna = False)
```

Out[281]:
```
0.0           38052
12000000.0       20
10000000.0       19
11000000.0       19
2000000.0        18
              ...  
36565280.0        1
439564.0          1
35610100.0        1
10217873.0        1
1413000.0         1
Name: revenue, Length: 6864, dtype: int64
```

In [282]:
```
df.revenue = df.revenue.replace(0, np.nan)
```

In [283]:
```
df.revenue = df.revenue.div(1000000)
```

In [284]:
```
df.rename(columns = {"revenue": "revenue_musd", "budget": "budget_musd"},
```

In [285]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   belongs_to_collection  4491 non-null   object 
 1   budget_musd            8890 non-null   float64
 2   genres                 43024 non-null  object 
 3   id                     45466 non-null  object 
 4   original_language      45455 non-null  object 
 5   overview               44512 non-null  object 
 6   popularity             45461 non-null  object 
 7   poster_path            45080 non-null  object 
 8   production_companies   33585 non-null  object 
 9   production_countries   39178 non-null  object 
 10  release_date           45379 non-null  object 
 11  revenue_musd           7408 non-null   float64
 12  runtime                45203 non-null  float64
 13  spoken_languages       41508 non-null  object 
 14  status                 45379 non-null  object 
 15  tagline                20412 non-null  object 
 16  title                  45460 non-null  object 
 17  vote_average           45460 non-null  float64
 18  vote_count             45460 non-null  float64
dtypes: float64(5), object(14)
memory usage: 6.6+ MB
```

Cleanig Numerical Columns (Part 2)

In [286]:
```
df.runtime.value_counts(dropna = False).head(20)
```

Out[286]:
```
90.0     2556
0.0      1558
100.0    1470
95.0     1412
93.0     1214
96.0     1104
92.0     1080
94.0     1062
91.0     1057
88.0     1032
97.0     1027
85.0     1024
98.0     1019
105.0    1002
89.0      958
87.0      919
110.0     850
86.0      846
99.0      794
102.0     791
Name: runtime, dtype: int64
```

In [287]:
```
df.runtime = df.runtime.replace(0, np.nan)
```

In [288]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   belongs_to_collection  4491 non-null   object 
 1   budget_musd            8890 non-null   float64
 2   genres                 43024 non-null  object 
 3   id                     45466 non-null  object 
 4   original_language      45455 non-null  object 
 5   overview               44512 non-null  object 
 6   popularity             45461 non-null  object 
 7   poster_path            45080 non-null  object 
 8   production_companies   33585 non-null  object 
 9   production_countries   39178 non-null  object 
 10  release_date           45379 non-null  object 
 11  revenue_musd           7408 non-null   float64
 12  runtime                43645 non-null  float64
 13  spoken_languages       41508 non-null  object 
 14  status                 45379 non-null  object 
 15  tagline                20412 non-null  object 
 16  title                  45460 non-null  object 
 17  vote_average           45460 non-null  float64
 18  vote_count             45460 non-null  float64
dtypes: float64(5), object(14)
memory usage: 6.6+ MB
```

In [5]:
```
pd.to_numeric(df.id)
```

In [290]:
```
df.id = pd.to_numeric(df.id, errors = 'coerce')
```

In [291]:
```
df.id.value_counts(dropna = False).head(20)
```

Out[291]:
```
NaN         3
141971.0    3
11115.0     2
25541.0     2
15028.0     2
132641.0    2
84198.0     2
13209.0     2
77221.0     2
152795.0    2
12600.0     2
10991.0     2
42495.0     2
14788.0     2
18440.0     2
168538.0    2
105045.0    2
159849.0    2
22649.0     2
4912.0      2
Name: id, dtype: int64
```

In [292]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   belongs_to_collection  4491 non-null   object 
 1   budget_musd            8890 non-null   float64
 2   genres                 43024 non-null  object 
 3   id                     45463 non-null  float64
 4   original_language      45455 non-null  object 
 5   overview               44512 non-null  object 
 6   popularity             45461 non-null  object 
 7   poster_path            45080 non-null  object 
 8   production_companies   33585 non-null  object 
 9   production_countries   39178 non-null  object 
 10  release_date           45379 non-null  object 
 11  revenue_musd           7408 non-null   float64
 12  runtime                43645 non-null  float64
 13  spoken_languages       41508 non-null  object 
 14  status                 45379 non-null  object 
 15  tagline                20412 non-null  object 
 16  title                  45460 non-null  object 
 17  vote_average           45460 non-null  float64
 18  vote_count             45460 non-null  float64
dtypes: float64(6), object(13)
memory usage: 6.6+ MB
```

In [293]:
```
df.popularity
```

Out[293]:
```
0        21.946943
1        17.015539
2          11.7129
3         3.859495
4         8.387519
           ...    
45461     0.072051
45462     0.178241
45463     0.903007
45464     0.003503
45465     0.163015
Name: popularity, Length: 45466, dtype: object
```

In [2]:
```
pd.to_numeric(df.popularity)
```

In [295]:
```
df.popularity = pd.to_numeric(df.popularity, errors = "coerce")
```

In [296]:
```
df.popularity.value_counts(dropna = False).head(20)
```

Out[296]:
```
0.000000    66
0.000001    56
0.000308    43
0.000220    40
0.000844    38
0.000578    38
0.001177    38
0.002001    28
0.003013    21
0.001393    19
0.003530    19
0.036471    18
0.002353    18
0.000603    16
0.001586    15
0.004425    14
0.001021    13
0.000431    13
0.004706    12
0.001247    11
Name: popularity, dtype: int64
```

In [297]:
```
df.vote_count.value_counts(dropna = False).head(20)
```

Out[297]:
```
1.0     3264
2.0     3132
0.0     2899
3.0     2787
4.0     2480
5.0     2097
6.0     1747
7.0     1570
8.0     1359
9.0     1194
10.0    1171
11.0     944
12.0     859
13.0     733
14.0     700
15.0     674
16.0     601
17.0     554
18.0     497
20.0     463
Name: vote_count, dtype: int64
```

In [298]:
```
df.vote_average.value_counts(dropna = False).head(20)
```

Out[298]:
```
0.0    2998
6.0    2468
5.0    2001
7.0    1886
6.5    1722
6.3    1603
5.5    1381
5.8    1369
6.4    1350
6.7    1342
6.8    1324
6.1    1281
6.6    1263
6.2    1253
5.9    1196
5.3    1082
5.7    1046
6.9    1037
5.6    1006
7.3    1000
Name: vote_average, dtype: int64
```

In [299]:
```
df.loc[df.vote_count == 0, "vote_average"] = np.nan
```

In [300]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 19 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   belongs_to_collection 4491 non-null   object 
 1   budget_musd           8890 non-null   float64
 2   genres                43024 non-null  object 
 3   id                    45463 non-null  float64
 4   original_language     45455 non-null  object 
 5   overview              44512 non-null  object 
 6   popularity            45460 non-null  float64
 7   poster_path           45080 non-null  object 
 8   production_companies  33585 non-null  object 
 9   production_countries  39178 non-null  object 
 10  release_date          45379 non-null  object 
 11  revenue_musd          7408 non-null   float64
 12  runtime               43645 non-null  float64
 13  spoken_languages      41508 non-null  object 
 14  status                45379 non-null  object 
 15  tagline               20412 non-null  object 
 16  title                 45460 non-null  object 
 17  vote_average          42561 non-null  float64
 18  vote_count            45460 non-null  float64
dtypes: float64(7), object(12)
memory usage: 6.6+ MB
```

Cleanig Date Time Columns

In [301]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   belongs_to_collection  4491 non-null   object 
 1   budget_musd            8890 non-null   float64
 2   genres                 43024 non-null  object 
 3   id                     45463 non-null  float64
 4   original_language      45455 non-null  object 
 5   overview               44512 non-null  object 
 6   popularity             45460 non-null  float64
 7   poster_path            45080 non-null  object 
 8   production_companies   33585 non-null  object 
 9   production_countries   39178 non-null  object 
 10  release_date           45379 non-null  object 
 11  revenue_musd           7408 non-null   float64
 12  runtime                43645 non-null  float64
 13  spoken_languages       41508 non-null  object 
 14  status                 45379 non-null  object 
 15  tagline                20412 non-null  object 
 16  title                  45460 non-null  object 
 17  vote_average           42561 non-null  float64
 18  vote_count             45460 non-null  float64
dtypes: float64(7), object(12)
memory usage: 6.6+ MB
```

In [302]:
```
df.release_date
```

Out[302]:
```
0        1995-10-30
1        1995-12-15
2        1995-12-22
3        1995-12-22
4        1995-02-10
            ...    
45461           NaN
45462    2011-11-17
45463    2003-08-01
45464    1917-10-21
45465    2017-06-09
Name: release_date, Length: 45466, dtype: object
```

In [1]:
```
pd.to_datetime(df.release_date)
```

In [ ]:
```
df.release_date = pd.to_datetime(df.release_date, errors = 'coerce')
```

In [304]:
```
df.release_date.value_counts(dropna = False)
```

Out[304]:
```
2008-01-01    136
2009-01-01    121
2007-01-01    118
2005-01-01    111
2006-01-01    101
             ...
1957-09-26      1
1938-11-21      1
1936-08-19      1
2010-01-27      1
1917-10-21      1
Name: release_date, Length: 17337, dtype: int64
```

Cleaning Text / String Columns

In [305]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   belongs_to_collection  4491 non-null   object 
 1   budget_musd            8890 non-null   float64
 2   genres                 43024 non-null  object 
 3   id                     45463 non-null  float64
 4   original_language      45455 non-null  object 
 5   overview               44512 non-null  object 
 6   popularity             45460 non-null  float64
 7   poster_path            45080 non-null  object 
 8   production_companies   33585 non-null  object 
 9   production_countries   39178 non-null  object 
 10  release_date           45379 non-null  object 
 11  revenue_musd           7408 non-null   float64
 12  runtime                43645 non-null  float64
 13  spoken_languages       41508 non-null  object 
 14  status                 45379 non-null  object 
 15  tagline                20412 non-null  object 
 16  title                  45460 non-null  object 
 17  vote_average           42561 non-null  float64
 18  vote_count             45460 non-null  float64
dtypes: float64(7), object(12)
memory usage: 6.6+ MB
```

In [306]:
```
df.original_language.value_counts(dropna = False).head(50)
```

Out[306]:
```
en     32269
fr      2438
it      1529
ja      1350
de      1080
es       994
ru       826
hi       508
ko       444
zh       409
sv       384
pt       316
cn       313
fi       297
nl       248
da       225
pl       219
tr       150
cs       130
el       113
no       106
fa       101
hu       100
ta        78
th        76
he        67
sr        63
ro        57
te        45
ar        39
ml        36
xx        33
bn        29
hr        29
mr        25
is        24
et        24
tl        23
id        20
lv        18
ka        18
sl        17
uk        16
bs        14
ca        12
NaN       11
bg        10
ab        10
vi        10
sk         9
Name: original_language, dtype: int64
```

In [307]:
```
df.title
```

Out[307]:
```
0                        Toy Story
1                          Jumanji
2                 Grumpier Old Men
3                Waiting to Exhale
4      Father of the Bride Part II
                    ...           
45461                       Subdue
45462          Century of Birthing
45463                     Betrayal
45464             Satan Triumphant
45465                     Queerama
Name: title, Length: 45466, dtype: object
```

In [308]:
```
df.title.value_counts(dropna = False).head(20)
```

Out[308]:
```
Cinderella              11
Alice in Wonderland      9
Hamlet                   9
Les Misérables           8
Beauty and the Beast     8
Treasure Island          7
A Christmas Carol        7
The Three Musketeers     7
Blackout                 7
Home                     6
Macbeth                  6
The Journey              6
The Hunters              6
The Forest               6
Mother                   6
Countdown                6
Love                     6
The Stranger             6
Heidi                    6
Bluebeard                6
Name: title, dtype: int64
```

In [309]:
```
df.overview[0]
```

Out[309]:
```
"Led by Woody, Andy's toys live happily in his room until Andy's birthday bri
ngs Buzz Lightyear onto the scene. Afraid of losing his place in Andy's hear
t, Woody plots against Buzz. But when circumstances separate Buzz and Woody f
rom their owner, the duo eventually learns to put aside their differences."
```

In [310]:
```
df.overview.value_counts(dropna = False).head(20)
```

Out[310]:
```
NaN
954
No overview found.
133
No Overview
7

5
No movie overview available.
3
A few funny little novels about different aspects of life.
3
Recovering from a nail gun shot to the head and 13 months of coma, doctor Pek
ka Valinta starts to unravel the mystery of his past, still suffering from to
tal amnesia.
3
King Lear, old and tired, divides his kingdom among his daughters, giving gre
at importance to their protestations of love for him. When Cordelia, youngest
and most honest, refuses to idly flatter the old man in return for favor, he
banishes her and turns for support to his remaining daughters. But Goneril an
d Regan have no love for him and instead plot to take all his power from him.
In a parallel, Lear's loyal courtier Gloucester favors his illegitimate son E
dmund after being told lies about his faithful son Edgar. Madness and tragedy
befall both ill-starred fathers.
3
Adaptation of the Jane Austen novel.
3
Released
3
In a hospital, ten soldiers are being treated for a mysterious sleeping sickn
ess. In a story in which dreams can be experienced by others, and in which go
ddesses can sit casually with mortals, a nurse learns the reason why the pati
ents will never be cured, and forms a telepathic bond with one of them.
2
The ghost of a samurai's wife takes revenge on her husband.
2
East-Berlin, 1961, shortly after the erection of the Wall. Konrad, Sophie and
three of their friends plan a daring escape to Western Germany. The attempt i
s successful, except for Konrad, who remains behind. From then on, and for th
e next 28 years, Konrad and Sophie will attempt to meet again, in spite of th
e Iron Curtain. Konrad, who has become a reputed Astrophysicist, tries to tak
e advantage of scientific congresses outside Eastern Germany to arrange encou
nters with Sophie. But in a country where the political police, the Stasi, mo
nitors the moves of all suspicious people (such as Konrad's sister Barbara an
d her husband Harald), preserving one's privacy, ideals and self-respect beco
mes an exhausting fight, even as the Eastern block begins its long process of
disintegration.                                                 2
Funny, entertaining comedy with a few storylines. All of them have one thing
in common - a resort town of Rimini in Italy.
2
Two literary women compete for 20 years: one writes for the critics; the othe
r one, to get rich.
2
All your favorite Pokémon characters are back, and are joined for the first t
ime by the legendary Pokémon Celebi and Suicune, in this latest exciting Poké
mon adventure! In order to escape a greedy Pokémon hunter, Celebi must use th
e last of its energy to travel through time to the present day. Celebi brings
```

```
along Sammy, a boy who had been trying to protect it. Along with Ash, Pikach
u, and the rest of the gang, Sammy and Celebi encounter an enemy far more adv
anced than the hunter left behind in the past. This new enemy possesses a Pok
éball called a "Dark Ball," which transforms the Pokémon it captures into evi
l and far stronger creatures. When Celebi is captured, the fate of the entire
forest is threatened. Let POKÉMON 4EVER transport you to a world of adventure
as Ash, Suicune and the rest take action to save the day!      2
Ten years into a marriage, the wife is disappointed by the husband's lack of
financial success, meaning she has to work and can't treat herself and the hu
sband finds the wife slovenly and mean-spirited: she neither cooks not cleans
particularly well and is generally disagreeable. In turn, he alternately igno
res her and treats her as a servant. Neither is particularly happy, not helpe
d by their unsatisfactory lodgers. The husband is easily seduced by an ex-col
league, a widow with a small child who needs some security, and considers lea
ving his wife.
2
Mary, a writer working on a novel about a love triangle, is attracted to her
publisher. Her suitor Jimmy is determined to break them up; he introduces Mar
y to the publisher's wife without telling Mary who she is.
2
Count de Chagnie has discovered Christine's singing talent on a market place
and sent her to his friend Carriere, the director of the Parisian opera. Howe
ver just when she arrives Carriere's dismissed. His arrogant successor refuse
s to let a woman of low birth sing in his opera, but graciously employs Chris
tine as gadrobiere for his wife Charlotta, who's installed as first singer. H
e also fights the phantom, an unknown guy who lives since many years in the c
atacombs below the opera and was granted privileges by Carriere. However the
phantom knows how to defend himself and at the same time helps Christine to h
er career.
2
As an ex-gambler teaches a hot-shot college kid some things about playing car
ds, he finds himself pulled into the world series of poker, where his protégé
is his toughest competition.
2
Name: overview, dtype: int64
```

In [311]:
```python
df.overview.replace("No overview found.", np.nan, inplace = True)
```

In [312]:
```python
df.overview.replace("No Overview", np.nan, inplace = True)
```

In [313]:
```python
df.overview.replace("No movie overview avilable.", np.nan, inplace = True)
```

In [314]:
```python
df.overview.replace(" ", np.nan, inplace = True)
```

In [315]:
```python
df.overview.replace("No overview yet.", np.nan, inplace = True)
```

In [316]:
```
df.overview.value_counts()
```

Out[316]:
```
King Lear, old and tired, divides his kingdom among his daughters, giving gre
at importance to their protestations of love for him. When Cordelia, youngest
and most honest, refuses to idly flatter the old man in return for favor, he
banishes her and turns for support to his remaining daughters. But Goneril an
d Regan have no love for him and instead plot to take all his power from him.
In a parallel, Lear's loyal courtier Gloucester favors his illegitimate son E
dmund after being told lies about his faithful son Edgar. Madness and tragedy
befall both ill-starred fathers.
3
A few funny little novels about different aspects of life.
3
Released
3
Adaptation of the Jane Austen novel.
3
Recovering from a nail gun shot to the head and 13 months of coma, doctor Pek
ka Valinta starts to unravel the mystery of his past, still suffering from to
tal amnesia.
3


..
"Of Time and The City" is both a love song and a eulogy to the director's bir
thplace of Liverpool, England. It is also a response to memory, reflection an
d the experience of losing a sense of place as the skyline changes and time t
akes it toll. The visual content of the film consists largely of archival cli
ps of Liverpool from the 1940s to the 1960s, their nostalgic charm darkened b
y accompanying music and by the counterpoint of Davies’ dry, at times dyspept
ic, voice-over narration. His voice thickens with emotion as he recalls the d
elights of juvenile movie-going or the ritual of a holiday trip to New Bright
on, across the River Mersey, and hardens with contempt when he turns his gaze
on the hoopla surrounding Queen Elizabeth’s coronation in 1953.    1
We are in the year 1871. A journalist for Versailles Television broadcasts a
soothing and official view of events while a Commune television is set up to
provide the perspectives of the Paris rebels. On a stage-like set, more than
200 actors interpret characters of the Commune, especially the Popincourt nei
ghbourhood in the XIth arrondissement. They voice their own thoughts and feel
ings concerning the social and political reforms. The scenes consist mainly o
f long camera takes.
1
A kindergarten director Troshkin is a dead ringer for a criminal nicknamed "D
ocent" who stole the priceless...
1
A man wakes up deep inside a cave. Suffering amnesia, he has no recollection
of how he came to be here or of what happened to the man whose body he finds
beside him. Tailed by a mysterious creature, he must continue through this st
range and fantastic world. Enclosed, Tolbiac has no other option to reach the
surface than to use REZO ZERO, secret observing cells in this cemetery-like a
bandoned mine.
1
50 years after decriminalisation of homosexuality in the UK, director Daisy A
squith mines the jewels of the BFI archive to take us into the relationships,
desires, fears and expressions of gay men and women in the 20th century.
1
Name: overview, Length: 44303, dtype: int64
```

In [317]:
```
df.tagline.value_counts(dropna = False).head(50)
```

Out[317]:
```
NaN
25054
Based on a true story.
7
Trust no one.
4
Be careful what you wish for.
4
-
4
Classic Albums
3
Some doors should never be opened.
3
A Love Story
3
Drama
3
Know Your Enemy
3
Which one is the first to return - memory or the murderer?
3
How far would you go?
3
The end is near.
3
There is no turning back
3
There are two sides to every love story.
3
Documentary
3
Who is John Galt?
3
Revenge Has No Limits
2
Who's next?
2
It's never too late.
2
Worlds Collide
2
Some things are better left top secret.
2
The Awakening
2
Nothing stays buried forever.
2
Every second counts.
2
Love never dies.
2
Based on a true story
2
Der deutsche Millionen-Film!
2
Once upon a time...
```

```
2
The band you know. The story you don't.
2
There's one in all of us.
2
From the very beginning, they knew they'd be friends to the end. What they di
dn't count on was everything in between.        2
No one stays innocent forever.
2
Every woman who has loved will understand
2
The hunt is on.
2
Two Films. One Love.
2
The first to die were the lucky ones!
2
Touched by Genius. Cursed by Madness. Blinded by Love.
2
How can you believe your eyes when they're not yours?
2
Evil will rise.
2
What you know about fear... doesn't even come close.
2
There is no solitude greater than that of the Samurai
2
Run for your life
2
You never forget your first love.
2
Terror runs deep.
2
Trust No One
2
The timeless tale of a special place where magic, hope and love grow.
2
Something wicked this way comes.
2
A love, a hope, a wall.
2
RELENTLESS SUSPENSE!
2
Name: tagline, dtype: int64
```

In [318]:

```python
df.tagline.replace("-", np.nan, inplace = True)
```

In [319]:
```python
df.tagline.value_counts()
```

Out[319]:
```
Based on a true story.                                          7
Trust no one.                                                   4
Be careful what you wish for.                                   4
Know Your Enemy                                                 3
Who is John Galt?                                               3
                                                               ..
A special force in a special kind of hell!                      1
Play it. Sing it. Shout it. Feel it.                            1
If It's On TV, It Must Be The Truth.                            1
"I LOVE YOU BABY, BUT MY WIFE JUST REFUSES TO UNDERSTAND!"      1
A deadly game of wits.                                          1
Name: tagline, Length: 20282, dtype: int64
```

Removing Duplicates

In [320]:
```python
df[df.duplicated(keep = False)].sort_values(by ='id')
```

Out[320]:

	belongs_to_collection	budget_musd	genres	id
7345	NaN	NaN	Crime|Drama|Thriller	5511.0
9165	NaN	NaN	Crime|Drama|Thriller	5511.0
24844	NaN	NaN	Comedy|Drama	11115.0
14012	NaN	NaN	Comedy|Drama	11115.0

In [321]:
```python
df.drop_duplicates(inplace = True)
```

In [322]:
```python
df[df.duplicated(subset = 'id', keep = False)].sort_values(by = 'id')
```

Out[322]:

	belongs_to_collection	budget_musd	genres	id
33826	NaN	30.000000	Comedy|Crime|Drama|Romance|Thriller	4912.0
5865	NaN	30.000000	Comedy|Crime|Drama|Romance|Thriller	4912.0
4114	Pokémon Collection	16.000000	Adventure|Fantasy|Animation|Action|Family	10991.0
44821	Pokémon Collection	16.000000	Adventure|Fantasy|Animation|Action|Family	10991.0

In [323]:
```python
df.drop_duplicates(subset = 'id', inplace = True)
```

In [324]:
```python
df.id.value_counts(dropna = False)
```

Out[324]:
```
862.0       1
74458.0     1
296206.0    1
107308.0    1
16247.0     1
           ..
44399.0     1
10138.0     1
32084.0     1
42191.0     1
461257.0    1
Name: id, Length: 45434, dtype: int64
```

Handling Missing Values & Removing Observation

In [325]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 45434 entries, 0 to 45465
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   belongs_to_collection  4488 non-null   object 
 1   budget_musd            8880 non-null   float64
 2   genres                 42992 non-null  object 
 3   id                     45433 non-null  float64
 4   original_language      45423 non-null  object 
 5   overview               44333 non-null  object 
 6   popularity             45430 non-null  float64
 7   poster_path            45048 non-null  object 
 8   production_companies   33562 non-null  object 
 9   production_countries   39151 non-null  object 
 10  release_date           45347 non-null  object 
 11  revenue_musd           7398 non-null   float64
 12  runtime                43615 non-null  float64
 13  spoken_languages       41480 non-null  object 
 14  status                 45349 non-null  object 
 15  tagline                20397 non-null  object 
 16  title                  45430 non-null  object 
 17  vote_average           42534 non-null  float64
 18  vote_count             45430 non-null  float64
dtypes: float64(7), object(12)
memory usage: 6.9+ MB
```

In [326]:
```
df.isna().sum()
```

Out[326]:
```
belongs_to_collection    40946
budget_musd              36554
genres                    2442
id                           1
original_language           11
overview                  1101
popularity                   4
poster_path                386
production_companies     11872
production_countries      6283
release_date                87
revenue_musd             38036
runtime                   1819
spoken_languages          3954
status                      85
tagline                  25037
title                        4
vote_average              2900
vote_count                   4
dtype: int64
```

In [327]:
```
df[df.title.isna()]
```

Out[327]:

	belongs_to_collection	budget_musd	genres	id	original_languag
19729	NaN	NaN	Action|Thriller|Drama	82663.0	e
19730	NaN	NaN	Carousel Productions|Vision View Entertainment...	NaN	104.
29502	Mardock Scramble Collection	NaN	Animation|Science Fiction	122662.0	j
35586	NaN	NaN	TV Movie|Action|Horror|Science Fiction	249260.0	e

In [328]:
```
df.dropna(subset = ['id','title'], inplace = True)
```

In [329]:
```
df.id = df.id.astype("int")
```

In [330]:
```
df.notna().sum(axis =1).value_counts().sort_values(ascending = False)
```

Out[330]:
```
15    12522
16    11455
14     5423
17     4265
18     3859
13     3041
12     1890
19     1132
11     1020
10      511
9       184
8       104
7        20
6         4
dtype: int64
```

In [331]:
```
df[df.notna().sum(axis = 1) == 7]
```

Out[331]:

	belongs_to_collection	budget_musd	genres	id	original_language	
2140	NaN	NaN	NaN	77314	fr	
4130	NaN	NaN	Drama|Thriller|Romance	109472	en	
14890	NaN	NaN	NaN	174748	no	
18572	NaN	NaN	Documentary	404471	fi	
19955	NaN	NaN	NaN	397339	en	
20301	NaN	NaN	NaN	367678	en	Dc
22798	NaN	NaN	NaN	158517	en	(
24157	NaN	NaN	NaN	287831	en	F a in
29309	NaN	NaN	NaN	335141	fr	
35652	NaN	NaN	NaN	374698	nl	
36421	NaN	NaN	NaN	382436	ru	
36524	NaN	NaN	NaN	166256	en	(A s
36550	NaN	NaN	NaN	9939	en	
37289	NaN	NaN	NaN	368128	en	ph
37640	NaN	NaN	NaN	54566	fi	
40082	NaN	NaN	NaN	411711	en	N
40203	NaN	NaN	NaN	410576	en	
41399	NaN	NaN	NaN	419289	es	
42573	NaN	NaN	NaN	440361	de	

	belongs_to_collection	budget_musd	genres	id	original_language
45070	NaN	NaN	NaN	439314	en

In [332]:
```
df.dropna(thresh = 10, inplace = True)
```

In [333]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 45118 entries, 0 to 45465
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   belongs_to_collection  4487 non-null   object 
 1   budget_musd            8878 non-null   float64
 2   genres                 42969 non-null  object 
 3   id                     45118 non-null  int32  
 4   original_language      45107 non-null  object 
 5   overview               44145 non-null  object 
 6   popularity             45118 non-null  float64
 7   poster_path            44886 non-null  object 
 8   production_companies   33561 non-null  object 
 9   production_countries   39147 non-null  object 
 10  release_date           45078 non-null  object 
 11  revenue_musd           7398 non-null   float64
 12  runtime                43552 non-null  float64
 13  spoken_languages       41462 non-null  object 
 14  status                 45052 non-null  object 
 15  tagline                20396 non-null  object 
 16  title                  45118 non-null  object 
 17  vote_average           42460 non-null  float64
 18  vote_count             45118 non-null  float64
dtypes: float64(6), int32(1), object(12)
memory usage: 6.7+ MB
```

In [334]:
```
df.isna().sum()
```

Out[334]:
```
belongs_to_collection    40631
budget_musd              36240
genres                    2149
id                           0
original_language           11
overview                   973
popularity                   0
poster_path                232
production_companies     11557
production_countries      5971
release_date                40
revenue_musd             37720
runtime                   1566
spoken_languages          3656
status                      66
tagline                  24722
title                        0
vote_average              2658
vote_count                   0
dtype: int64
```

Final(Cleaning) steps

In [335]:
```
df.status.value_counts()
```

Out[335]:
```
Released           44691
Rumored              226
Post Production       98
In Production         20
Planned               15
Canceled               2
Name: status, dtype: int64
```

In [336]:
```
df = df.loc[df.status == "Released"].copy()
```

In [337]:
```
df
```

Out[337]:

	belongs_to_collection	budget_musd	genres	id	original_language
0	Toy Story Collection	30.0	Animation|Comedy|Family	862	en
1	NaN	65.0	Adventure|Fantasy|Family	8844	en
2	Grumpy Old Men Collection	NaN	Romance|Comedy	15602	en
3	NaN	16.0	Comedy|Drama|Romance	31357	en
4	Father of the Bride Collection	NaN	Comedy	11862	en
...	...	...	...	...	...
45461	NaN	NaN	Drama|Family	439050	fa
45462	NaN	NaN	Drama	111109	tl
45463	NaN	NaN	Action|Drama|Thriller	67758	en
45464	NaN	NaN	NaN	227506	en
45465	NaN	NaN	NaN	461257	en

44691 rows × 19 columns

In [338]:
```
df.drop(columns = ["status"], inplace = True)
```

In [339]:
```
col = ["id","title","tagline","release_date","genres","belongs_to_collecti
        "original_language","budget_musd","revenue_musd","production_compan
       "vote_count","vote_average","popularity", "runtime","overview", "spo
```

In [340]:
```
df = df.loc[:,col]
```

In [341]:
```
df.reset_index(drop = True, inplace = True)
```

In [342]:
```
df.poster_path[0]
```

Out[342]:
```
'/rhIRbceoE9lR4veEXuwCC2wARtG.jpg'
```

In [343]:
```
base_poster_url = "http://image.tmdb.org/t/p/w185/"
```

In [344]:
```
df.poster_path = "<img src='" + base_poster_url + df.poster_path + "' styl
```

In [345]:
```
#df.to_csv("movies_clean.csv", index = False)
```

In [346]:
```
#pd.read_csv("movies_clean.csv").info()
```

In [ ]:
```

```

In [ ]:
```

```