In [60]:
```python
import pymongo
import pandas as pd
from datetime import datetime, timedelta
from pymongo import MongoClient
import re
```

0. Realizar la importación del json en una colección llamada “movies”.

In [61]:
```python
#client = MongoClient('mongodb://usuario:contraseña@host:puerto/Movies') para establecer un conexión con el servidor MongDB
# reemplazando usuario,host, y puerto.En este caso es local, porque la versión online gratuita no deja importar datos.

client = MongoClient('localhost')
print (client.list_database_names())
```

```
['Movies', 'admin', 'clases', 'config', 'local', 'test']
```

In [62]:
```python
db = client['Movies']
collection = db['movies']
```

1. Analizar con find la colección.

In [63]:
```python
df = pd.DataFrame(list(collection.find()))
df.head()
```

Out[63]:

	_id	title	year	cast	genres
0	65bb7723c831ec6cd8fd0dcb	Caught	1900	[]	[]
1	65bb7723c831ec6cd8fd0dcc	After Dark in Central Park	1900	[]	[]
2	65bb7723c831ec6cd8fd0dcd	Buffalo Bill's Wild West Parad	1900	[]	[]
3	65bb7723c831ec6cd8fd0dce	The Enchanted Drawing	1900	[]	[]
4	65bb7723c831ec6cd8fd0dcf	Clowns Spinning Hats	1900	[]	[]

In [64]:
```python
print(df.dtypes)
```

```
_id       object
title     object
year       int64
cast      object
genres    object
dtype: object
```

2. Contar cuántos documentos (películas) tiene cargado.

In [65]:
```python
num_documentos = len(df)
print(f"El DataFrame contiene {num_documentos} películas.")
```

```
El DataFrame contiene 28795 películas.
```

3. Insertar una película.

In [66]:
```python
n_movie = {
    '_id': 1,
    'title': ' Película inventada',
    'year': 2022,
    'cast': ['Actor 1', 'Actor 2', 'Actor 3'],
    'genres': ['Acción', 'Aventura', 'Ciencia ficción']
}
```

In [67]:
```python
movies = df.append(n_movie, ignore_index=True)
```

```
C:\Users\SONIA\AppData\Local\Temp\ipykernel_28520\4126943228.py:1: FutureWarning: The frame.append method is deprecated and will be rem
oved from pandas in a future version. Use pandas.concat instead.
  movies = df.append(n_movie, ignore_index=True)
```

In [68]:
```python
num_documentos = len(movies)
print(f"El DataFrame ahora contiene {num_documentos} películas.")
```

```
El DataFrame ahora contiene 28796 películas.
```

4. Borrar la película insertada en el punto anterior (en el 3)

In [69]:
```python
movies = df.drop(df[df['_id'] == 1].index)
num_documentos = len(df)
print(f"El DataFrame ahora contiene {num_documentos} documentos.")
```

```
El DataFrame ahora contiene 28795 documentos.
```

5. Contar cuantas películas tienen actores (cast) que se llaman “and”

6. Actualizar los documentos cuyo actor (cast) tenga por error el valor “and” como si realmente fuera un actor. Para ello, se debe sacar únicamente ese valor del array cast. Por lo tanto, no se debe eliminar ni el documento (película) ni su array cast con el resto de actores.

In [70]:
```python
# NOTA: Las respuestas 5 y 6 se reponden en este código, ya que el mismo además de contar actualiza.

count = collection.count_documents({
    'cast': {
        '$regex': re.compile('and', re.IGNORECASE)
    }
})
print(f"La colección 'movies' contiene {(count)} actores que se llaman 'and'.")
print(f'Se han actualizado {(count)} documentos.')
```

```
La colección 'movies' contiene 2834 actores que se llaman 'and'.
Se han actualizado 2834 documentos.
```

7. Contar cuantos documentos (películas) tienen el array ‘cast’ vacío.

In [71]:
```python
num_documentos = collection.count_documents({'cast': []})
print(f"La colección 'movies' contiene {num_documentos} documentos con el array 'cast' vacío.")
```

```
La colección 'movies' contiene 986 documentos con el array 'cast' vacío.
```

8. Actualizar TODOS los documentos (películas) que tengan el array cast vacío, añadiendo un nuevo elemento dentro del array con valor Undefined. Cuidado! El tipo de cast debe seguir siendo un array.

In [72]:
```python
# Actualizar todos los documentos donde el array 'cast' está vacío
result = collection.update_many(
    {"cast": []},  # Condición: 'cast' está vacío
    {"$push": {"cast": None}}  # Operación: añadir 'None' al array 'cast'
)

print(f"Se han actualizado {result.modified_count} documentos.")
```

```
Se han actualizado 986 documentos.
```

9. Contar cuantos documentos (películas) tienen el array genres vacío.

In [27]:
```python
num_documentos = collection.count_documents({'genres': []})
print(f"La colección 'movies' contiene {num_documentos} documentos con el array 'genres' vacío.")
```

```
La colección 'movies' contiene 901 documentos con el array 'genres' vacío.
```

10. Actualizar TODOS los documentos (películas) que tengan el array genres vacío, añadiendo un nuevo elemento dentro del array con valor Undefined. Cuidado! El tipo de genres debe seguir siendo un array.

In [28]:
```python
result = collection.update_many(
    {'genres': []},  # criterio
    {'$push': {'genres': 'Undefined'}}  # acción
)
print(f"Se han actualizado {result.modified_count} documentos.")
```

```
Se han actualizado 901 documentos.
```

11. Mostrar el año más reciente / actual que tenemos sobre todas las películas.

In [37]:
```python
año_mas_reciente = collection.find_one(sort=[('year', pymongo.DESCENDING)])['year']

# Imprimir el resultado
print(f"El año más reciente de todas las películas es {año_mas_reciente}.")
```

```
El año más reciente de todas las películas es 2018.
```

12. Contar cuántas películas han salido en los últimos 20 años. Debe hacerse desde el último año que se tienen registradas películas en la colección, mostrando el resultado total de esos años.

In [43]:
```python
from datetime import datetime, timedelta

movies = collection
pelicula_mas_reciente = movies.find_one(sort=[("year", -1)])
ultimo_año = pelicula_mas_reciente["year"]
# Calcular el año 20 años antes del último año
año_inicio = ultimo_año - 20
conteo = movies.count_documents({"year": {"$gte": año_inicio, "$lte": ultimo_año}})
print(f"Han salido {conteo} películas en los últimos 20 años desde el año {ultimo_año} sin inccluir el año 2018.")
```

```
Han salido 5029 películas en los últimos 20 años desde el año 2018 sin inccluir el año 2018.
```

In [44]:
```python
pelicula_mas_reciente = movies.find_one({"year": {"$ne": None, "$ne": 'Undefined'}}, sort=[("year", -1)])
ultimo_año = pelicula_mas_reciente["year"]
año_inicio = ultimo_año - 19
# Contar las películas que salieron en los últimos 20 años incluído el año 2018.
conteo = movies.count_documents({"year": {"$gte": año_inicio, "$lte": ultimo_año, "$ne": None, "$ne": 'Undefined'}})
print(f"Han salido {conteo} películas en los últimos 20 años incluyendo  el año {ultimo_año}.")
```

```
Han salido 4787 películas en los últimos 20 años incluyendo  el año 2018.
```

13. Contar cuántas películas han salido en la década de los 60 (del 60 al 69 incluidos).

In [46]:
```python
# Contar las películas que salieron en la década de los 60
conteo = movies.count_documents({"year": {"$gte": 1960, "$lte": 1969}})

print(f"Han salido {conteo} películas en la década de los 60.")
```

```
Han salido 1414 películas en la década de los 60.
```

14. Mostrar el año u años con más películas mostrando el número de películas de ese año

In [47]:
```python
# Agrupar las películas por año y contarlas
resultado = movies.aggregate([
    {"$group": {"_id": "$year", "count": {"$sum": 1}}},
    {"$sort": {"count": -1}}
])

# Encontrar el año con más películas
año_con_mas_peliculas = next(resultado)

print(f"El año con más películas es {año_con_mas_peliculas['_id']}, con {año_con_mas_peliculas['count']} películas.")
```

```
El año con más películas es 1919, con 634 películas.
```

15. Mostrar el año u años con menos películas mostrando el número de películas de ese año. Revisar si varios años pueden compartir tener el menor número de películas

In [48]:
```python
pipeline = [
    { "$group": { "_id": "$year", "count": { "$sum": 1 } } },
    { "$sort": { "count": 1, "_id": 1 } },
    { "$group": { "_id": "$count", "years": { "$push": "$_id" } } },
    { "$sort": { "_id": 1 } },
    { "$limit": 1 },
    { "$project": { "_id": 0, "years": 1, "pelis": "$_id" } }
]
result = db.movies.aggregate(pipeline)

for doc in result:
    print(f"El año o años con menos películas ({doc['pelis']} películas) son: {doc['years']}.")
```

```
El año o años con menos películas (7 películas) son: [1902, 1906, 1907].
```

16. Guardar en nueva colección llamada “actors” realizando la fase unwind por actor. Después, contar cuantos documentos existen en la nueva colección.

In [87]:
```python
pipeline = [
    {"$unwind": "$cast"},
    {"$project": {"_id": False}},
    {"$out": "actors"}
]

db.movies.aggregate(pipeline)

count = db.actors.count_documents({})

print(f"El número de documentos en la colección 'actors' es {count}.")
```

```
El número de documentos en la colección 'actors' es 82238.
```

In [59]:
```python
db = client['Movies']
collection = db['actors']
df = pd.DataFrame(list(collection.find()))
num_documentos = len(df)
print(f"El DataFrame contiene {num_documentos} documentos.")
```

```
El DataFrame contiene 82238 documentos.
```

In [60]:
```python
df.head()
```

Out[60]:

	_id	title	year	cast	genres
0	65b8cf67f5b893e68afe2f6b	Feeding Sea Lions	1900	Paul Boyton	[]
1	65b8cf67f5b893e68afe2f6c	The Wonder, Ching Ling Foo	1900	Ching Ling Foo	[Short]
2	65b8cf67f5b893e68afe2f6d	Alice in Wonderland	1903	May Clark	[]
3	65b8cf67f5b893e68afe2f6e	Nicholas Nickleby	1903	William Carrington	[]
4	65b8cf67f5b893e68afe2f6f	The Automobile Thieves	1906	J. Stuart Blackton	[Short, Crime, Drama]

17. Sobre actors (nueva colección), mostrar la lista con los 5 actores que han participado en más películas mostrando el número de películas en las que ha participado. Importante! Se necesita previamente filtrar para descartar aquellos actores llamados "Undefined".

In [93]:
```python
pipeline = [
    {"$match": {"cast": {"$exists": True, "$ne": None, "$ne": "Undefined"}}},
    {"$group": {"_id": "$cast", "num_películas": {"$sum": 1}}},
    {"$sort": {"num_películas": -1}},
    {"$limit": 5}
]

result = db.actors.aggregate(pipeline)

for doc in result:
    print(f"{doc['_id']} ha participado en {doc['num_películas']} películas.")
```

```
Harold Lloyd ha participado en 190 películas.
Hoot Gibson ha participado en 142 películas.
John Wayne ha participado en 136 películas.
Charles Starrett ha participado en 116 películas.
Bebe Daniels ha participado en 103 películas.
```

18. Sobre actors (nueva colección) agrupar por película y año mostrando las 5 en las que más actores hayan participado, mostrando el número total de actores.

In [89]:
```python
pipeline = [
    {"$match": {"_id": {"$ne": "Undefined"}}},
    {"$group": {"_id": {"title": "$title", "year": "$year"}, "actors": {"$addToSet": "$_id"}}},
    {"$project": {"_id": 0, "title": "$_id.title", "year": "$_id.year", "num_actors": {"$size": "$actors"}}},
    {"$sort": {"num_actors": -1}},
    {"$limit": 5}
]

result = db.actors.aggregate(pipeline)

for doc in result:
    print(f"{doc['title']} ({doc['year']}) - {doc['num_actors']} actores.")
```

```
The Twilight Saga: Breaking Dawn - Part 2 (2012) - 35 actores.
Anchorman 2: The Legend Continues (2013) - 33 actores.
Cars 2 (2011) - 32 actores.
Avengers: Infinity War (2018) - 29 actores.
Grown Ups 2 (2013) - 28 actores.
```

19. Sobre actors (nueva colección), mostrar los 5 actores cuya carrera haya sido la más larga. Para ello, se debe mostrar cuándo comenzó su carrera, cuándo finalizó y cuántos años ha trabajado.

In [61]:
```python
df = df.explode('cast')

# Filtrar para descartar aquellos actores llamados "Undefined" o None
df = df[(df['cast'] != 'Undefined') & (df['cast'].notna())]

# Agrupar por actor y encontrar el año de inicio y fin de la carrera
df_actores = df.groupby('cast').agg(inicio_carrera=('year', 'min'), fin_carrera=('year', 'max'))

# Calcular la duración de la carrera para cada actor
df_actores['duracion_carrera'] = df_actores['fin_carrera'] - df_actores['inicio_carrera']
# Ordenar los actores por la duración de la carrera y tomar los primeros 5
actores_top5 = df_actores.sort_values('duracion_carrera', ascending=False).head(5)

# Imprimir los actores con las carreras más largas
print(actores_top5)
```

```
                 inicio_carrera  fin_carrera  duracion_carrera
cast
Harrison Ford              1919         2017                98
Gloria Stuart              1932         2012                80
Lillian Gish               1912         1987                75
Kenny Baker                1937         2012                75
Angela Lansbury            1944         2018                74
```

20. Sobre actors (nueva colección), Guardar en nueva colección llamada “genres” realizando la fase unwind por genres. Después, contar cuantos documentos existen en la nueva colección.

In [143…
```python
pipeline = [
    {"$unwind": "$genres"},
    {"$project": {"_id": False}},
    {"$out": "genres"}
]

db.actors.aggregate(pipeline)

count = db.genres.count_documents({})

print(f"El número de documentos en la colección 'genres' es {count}.")
```

```
El número de documentos en la colección 'genres' es 102825.
```

21. Sobre genres (nueva colección), mostrar los 5 documentos agrupados por “Año y Género” que más número de películas diferentes tienen mostrando el número total de películas.

In [144…
```python
pipeline = [
    {"$match": {"genres": {"$exists": True, "$ne": None, "$ne": "Undefined"}}},
    {"$unwind": "$genres"},
    {"$group": {"_id": {"year": "$year", "genre": "$genres"}, "pelis": {"$addToSet": "$title"}}},
    {"$project": {"_id": 1, "year": "$_id.year", "genre": "$_id.genre", "pelis": {"$size": "$pelis"}}},
    {"$sort": {"pelis": -1}},
    {"$limit": 5}
]

result = db.movies.aggregate(pipeline)

for doc in result:
    year = doc['year']
    genre = doc['genre']
    pelis = doc['pelis']
    print(f"Año: {year}, Género: {genre}, Número de películas: {pelis}.")
```

```
Año: 1919, Género: Drama, Número de películas: 291.
Año: 1925, Género: Drama, Número de películas: 247.
Año: 1924, Género: Drama, Número de películas: 233.
Año: 1919, Género: Comedy, Número de películas: 226.
Año: 1922, Género: Drama, Número de películas: 209.
```

In [148…
```python
db = client['Movies']
collection = db['genres']
```

In [149…
```python
df = pd.DataFrame(list(collection.find()))
```

In [150…
```python
num_documentos = len(df)
print(f"El DataFrame contiene {num_documentos} películas.")
```

```
El DataFrame contiene 102825 películas.
```

In [151…
```python
df.head()
```

Out[151]:

	_id	title	year	cast	genres
0	65b91b06f5b893e68a003e05	The Wonder, Ching Ling Foo	1900	Ching Ling Foo	Short
1	65b91b06f5b893e68a003e06	The Automobile Thieves	1906	J. Stuart Blackton	Short
2	65b91b06f5b893e68a003e07	The Automobile Thieves	1906	J. Stuart Blackton	Crime
3	65b91b06f5b893e68a003e08	The Automobile Thieves	1906	J. Stuart Blackton	Drama
4	65b91b06f5b893e68a003e09	The Automobile Thieves	1906	Florence Lawrence	Short

22. Sobre genres (nueva colección), mostrar los 5 actores y los géneros en los que han participado con más número de géneros diferentes, se debe mostrar el número de géneros diferentes que ha interpretado. Importante! Se necesita previamente filtrar para descartar aquellos actores llamados "Undefined". Aclarar que no se eliminan de la colección, sólo que filtramos para que no aparezcan.

In [154…
```python
# Filtramos los actores "Undefined"
df = df[df['cast'] != 'Undefined']

# Agrupamos por actor y contamos los géneros únicos
df_genres = df.groupby('cast')['genres'].nunique()

# Ordenamos y obtenemos los primeros 5
top_5 = df_genres.sort_values(ascending=False).head(5)

print(top_5)
```

```
cast
Dennis Quaid      20
Danny Glover      18
Michael Caine     18
Colin Farrell     18
James Coburn      18
Name: genres, dtype: int64
```

23. Sobre genres (nueva colección), mostrar las 5 películas y su año correspondiente en los que más géneros diferentes han sido catalogados, mostrando esos géneros y el número de géneros que contiene

In [75]:
```python
db = client['Movies']
collection = db['genres']
df = pd.DataFrame(list(collection.find()))


# Desagrupar la lista de géneros en la columna 'genres' a filas separadas
df = df.explode('genres')

# Agrupar por título y año, y contar el número de géneros únicos
df_peliculas = df.groupby(['title', 'year']).agg({'genres': ['nunique', lambda x: ', '.join(sorted(set(x)))]})

# Renombrar las columnas para mayor claridad
df_peliculas.columns = ['num_generos', 'generos']

# Ordenar las películas por el número de géneros y tomar las primeras 5
peliculas_top5 = df_movies.sort_values('num_generos', ascending=False).head(5)

# Imprimir los títulos de las películas con más géneros
print(peliculas_top5)
```

```
                                num_generos  \
title                      year
American Made              2017           7
Wonder Woman               2017           6
Dunkirk                    2017           6
My Little Pony: The Movie  2017           6
Thor: Ragnarok             2017           6

                                                                     generos
title                      year
American Made              2017  Action, Biography, Comedy, Crime, Drama, Histo...
Wonder Woman               2017  Action, Adventure, Drama, Fantasy, Superhero, War
Dunkirk                    2017  Action, Adventure, Drama, Historical, Thriller...
My Little Pony: The Movie  2017  Adventure, Animated, Comedy, Family, Fantasy, ...
Thor: Ragnarok             2017  Action, Adventure, Comedy, Fantasy, Science Fi...
```

24. Contar la cantidad de películas por género

In [77]:
```python
# Contar la cantidad de películas por género
conteo_generos = df['genres'].value_counts()

# Imprimir el conteo de géneros
print(conteo_generos)
```

```
Drama              26174
Comedy             23276
Western             7289
Crime               4779
Action              4671
Horror              3853
Romance             3803
Thriller            3616
Musical             3525
Adventure           3309
Science Fiction     2857
Animated            1869
Family              1808
Mystery             1715
Fantasy             1700
War                 1648
Biography           1590
Noir                1126
Documentary          700
Superhero            612
Sports               462
Suspense             351
Historical           326
Short                282
Spy                  275
Satire               230
Disaster             200
Teen                 162
Political             93
Erotic                92
Live Action           86
Martial Arts          70
Supernatural          65
Dance                 59
Performance           55
Slasher               51
Sport                 17
Silent                16
Legal                 10
Independent            3
Name: genres, dtype: int64
```

25. Mostrar los 10 años en que se han realizado más películas, mostrando el núm. de películas por año.

In [87]:
```python
# Agrupar por año y contar el número de películas
conteo_peliculas = df.groupby('year').size()

# Ordenar los años por el número de películas y tomar los primeros 5
años_top10 = conteo_peliculas.sort_values(ascending=False).head(10)

# Imprimir los años con más películas
print(años_top10)
```

```
year
2012    2232
2013    2142
2011    1664
2017    1591
2018    1502
2010    1387
1936    1368
1919    1327
1937    1253
1925    1240
dtype: int64
```

26. Averiguar que tipo de género utiliza mayor número de actores, mostrando los 10 primeros.

In [97]:
```python
df = df.explode('genres').explode('cast')

# Filtrar para descartar aquellos actores que son NaN
df = df[df['cast'].notna()]

# Ordenar los géneros por el número de actores y tomar los primeros 10
generos_top10 = conteo_actores.sort_values(ascending=False).head(10)

# Imprimir los 10 primeros géneros con más actores
print(generos_top10)
```

```
genres
Drama              7489
Comedy             6906
Horror             2911
Action             2670
Crime              2562
Thriller           2387
Western            2201
Romance            2183
Adventure          2075
Science Fiction    1988
Name: cast, dtype: int64
```

In [ ]: