In [71]:

```python
import requests
import pandas as pd

url = "https://moviesdatabase.p.rapidapi.com/actors"
headers = {
    "X-RapidAPI-Key": "9fa87ae34emsh635508e9c0183d7p1297a3jsnb0600b14118e",
    "X-RapidAPI-Host": "moviesdatabase.p.rapidapi.com"
}

all_data = []  # To store all data from pages

# Loop over pages 1 to 50
for page_num in range(1, 51):
    querystring = {"page": str(page_num)}
    response = requests.get(url, headers=headers, params=querystring)

    if response.status_code == 200:
        data = response.json()
        if 'results' in data:
            all_data.extend(data['results'])  # Append results to the list

# Convert the list of dictionaries into a DataFrame
df = pd.DataFrame(all_data)

# Display the DataFrame
df
```

Out[71]:

	_id	nconst	primaryName	birthYear	deathYear	primaryProfession	knownForTitles
0	617082fa2299cb9f2c2717ac	nm0000001	Fred Astaire	1899	1987	soundtrack,actor,miscellaneous	tt0050419,tt0072308,tt0031983,tt0053137
1	617082fa2299cb9f2c2717ad	nm0000005	Ingmar Bergman	1918	2007	writer,director,actor	tt0083922,tt0069467,tt0050986,tt0050976
2	617082fa2299cb9f2c2717ae	nm0000006	Ingrid Bergman	1915	1982	actress,soundtrack,producer	tt0034583,tt0038787,tt0038109,tt0036855
3	617082fa2299cb9f2c2717af	nm0000007	Humphrey Bogart	1899	1957	actor,soundtrack,producer	tt0043265,tt0037382,tt0042593,tt0034583
4	617082fa2299cb9f2c2717b0	nm0000008	Marlon Brando	1924	2004	actor,soundtrack,director	tt0047296,tt0068646,tt0070849,tt0078788
...	...	...	...	...	...	...	...
495	617082fa2299cb9f2c27199b	nm0000497	Jennifer Lien	1974	\N	actress,producer,script_department	tt0112178,tt0133189,tt0120586,tt0106100
496	617082fa2299cb9f2c27199c	nm0000498	Matthew Lillard	1970	\N	actor,producer,soundtrack	tt0331632,tt0267913,tt0133189,tt1033575
497	617082fa2299cb9f2c27199d	nm0000499	Bai Ling	\N	\N	actress,producer,writer	tt1121931,tt0109506,tt0119994,tt0405336
498	617082fa2299cb9f2c27199e	nm0000500	Richard Linklater	1960	\N	producer,director,writer	tt0243017,tt1065073,tt0405296,tt2209418
499	617082fa2299cb9f2c27199f	nm0000502	Christopher Lloyd	1938	\N	actor,soundtrack,producer	tt0106220,tt0088763,tt0101272,tt0096438

500 rows × 7 columns

In [68]:

```python
import requests
import pandas as pd

# Initialize an empty list to store data from all pages
all_data = []

# Loop over pages 1 to 50
for page_num in range(1, 51):
    url = "https://moviesdatabase.p.rapidapi.com/titles"
    querystring = {"page": str(page_num)}
    headers = {
        "X-RapidAPI-Key": "9fa87ae34emsh635508e9c0183d7p1297a3jsnb0600b14118e",
        "X-RapidAPI-Host": "moviesdatabase.p.rapidapi.com"
    }

    response = requests.get(url, headers=headers, params=querystring)

    if response.status_code == 200:
        data = response.json()
        if 'results' in data:
            results_data = data['results']
            all_data.extend(results_data)  # Append results to the list
        else:
            print(f"No results found for page {page_num}")
    else:
        print(f"Failed to fetch data for page {page_num}. Status code: {response.status_code}")

# Convert the list of dictionaries into a DataFrame
df = pd.DataFrame(all_data)

# Selecting specific columns
new_df = df[['_id', 'id', 'titleText', 'releaseYear', 'releaseDate']]

# Displaying the new DataFrame
new_df

```

Out[68]:

	_id	id	titleText	releaseYear	releaseDate
0	61e57fd65c5338f43c777f4a	tt0000081	{'text': 'Les haleurs de bateaux', '__typename...	{'year': 1896, 'endYear': None, '__typename': ...	None
1	61e57fd65c5338f43c777f4c	tt0000045	{'text': 'Les blanchisseuses', '__typename': '...	{'year': 1896, 'endYear': None, '__typename': ...	{'day': None, 'month': None, 'year': 1896, '__...
2	61e57fd65c5338f43c777f4e	tt0000066	{'text': 'Dessinateur: Von Bismark', '__typena...	{'year': 1896, 'endYear': None, '__typename': ...	None
3	61e57fd65c5338f43c777f50	tt0000049	{'text': 'Boxing Match; or, Glove Contest', '_...	{'year': 1896, 'endYear': None, '__typename': ...	{'day': None, 'month': 1, 'year': 1896, '__typ...
4	61e57fd65c5338f43c777f52	tt0000103	{'text': 'Plus fort que le maître', '__typenam...	{'year': 1896, 'endYear': None, '__typename': ...	None
...	...	...	...	...	...
495	61e57fd7d8a2362a33777f8e	tt0000234	{'text': 'Cléopâtre', '__typename': 'TitleText'}	{'year': 1899, 'endYear': None, '__typename': ...	{'day': None, 'month': None, 'year': 1899, '__...
496	61e57fd7d8a2362a33777f90	tt0000276	{'text': 'Bataille d'oreillers', '__typename':...	{'year': 1900, 'endYear': None, '__typename': ...	None
497	61e57fd7d8a2362a33777f92	tt0000566	{'text': 'Professorens Morgenavis', '__typenam...	{'year': 1906, 'endYear': None, '__typename': ...	{'day': 24, 'month': 3, 'year': 1906, '__typen...
498	61e57fd7d8a2362a33777f94	tt0000328	{'text': 'La petite magicienne', '__typename':...	{'year': 1900, 'endYear': None, '__typename': ...	None
499	61e57fd7d8a2362a33777f96	tt0000488	{'text': 'The Land Beyond the Sunset', '__type...	{'year': 1912, 'endYear': None, '__typename': ...	{'day': 28, 'month': 10, 'year': 1912, '__type...

500 rows × 5 columns

In [74]:
```
df.to_csv('actor.csv', index=False)
print("DataFrame successfully saved to 'actor.csv'.")
```
```
DataFrame successfully saved to 'actor.csv'.
```

In [75]:
```
new_df.to_csv('movie_title.csv', index=False)
print("DataFrame successfully saved to 'movie_title.csv'.")
```
```
DataFrame successfully saved to 'movie_title.csv'.
```

In [78]:
```python
import pandas as pd
from ydata_profiling import ProfileReport

file_path = "actor.csv"
df = pd.read_csv(file_path)

profile = ProfileReport(df, title='Pandas Profiling Report', explorative=True)
profile.to_file("actor.html")
```

```
Summarize dataset: 100%|██████████████████████████████████████████████████| 16/16 [00:00<00:00, 32.31it/s, Completed]
Generate report structure: 100%|█████████████████████████████████████████████████████████| 1/1 [00:03<00:00,  3.39s/it]
Render HTML: 100%|███████████████████████████████████████████████████████████████████████| 1/1 [00:00<00:00,  3.52it/s]
Export report to file: 100%|████████████████████████████████████████████████████████████| 1/1 [00:00<00:00, 140.46it/s]
```

In [79]:
```python
import pandas as pd
from ydata_profiling import ProfileReport

file_path = "movie_title.csv"
df = pd.read_csv(file_path)

profile = ProfileReport(df, title='Pandas Profiling Report', explorative=True)
profile.to_file("title.html")
```

```
Summarize dataset: 100%|██████████████████████████████████████████████████| 14/14 [00:00<00:00, 36.77it/s, Completed]
Generate report structure: 100%|█████████████████████████████████████████████████████████| 1/1 [00:02<00:00,  2.02s/it]
Render HTML: 100%|███████████████████████████████████████████████████████████████████████| 1/1 [00:00<00:00,  5.08it/s]
Export report to file: 100%|████████████████████████████████████████████████████████████| 1/1 [00:00<00:00, 236.85it/s]
```