# Analysis of IMDB Data

We will analyze a subset of IMDB's actors, genres, movie actors, and movie ratings data. This dataset comes to us from Kaggle (https://www.kaggle.com/datasets/ashirwadsangwan/imdb-dataset) although we have taken steps to pull this data into a public S3 bucket:

- s3://cis9760-lecture9-movieanalysis/name.basics.new.tsv ---> Name Basics
- s3://cis9760-lecture9-movieanalysis/title.basic.new.tsv ---> Title Basics
- s3://cis9760-lecture9-movieanalysis/title.principles.new.tsv ---> Title Principles
- s3://cis9760-lecture9-movieanalysis/title.ratings.new.tsv ---> Title Ratings

# Content

**name.basics.tsv.gz – Contains the following information for names:**
nconst (string) - alphanumeric unique identifier of the name/person.
primaryName (string)– name by which the person is most often credited.
birthYear – in YYYY format.
deathYear – in YYYY format if applicable, else .
primaryProfession (array of strings)– the top-3 professions of the person.
knownForTitles (array of tconsts) – titles the person is known for.

**title.basics.tsv.gz - Contains the following information for titles:**
tconst (string) - alphanumeric unique identifier of the title.
titleType (string) – the type/format of the title (e.g. movie, short, tvseries, tvepisode, video, etc).
primaryTitle (string) – the more popular title / the title used by the filmmakers on promotional materials at the point of release.
originalTitle (string) - original title, in the original language.
isAdult (boolean) - 0: non-adult title; 1: adult title.
startYear (YYYY) – represents the release year of a title. In the case of TV Series, it is the series start year.
endYear (YYYY) – TV Series end year. for all other title types.
runtimeMinutes – primary runtime of the title, in minutes.
genres (string array) – includes up to three genres associated with the title.

**title.principals.tsv – Contains the principal cast/crew for titles:**
tconst (string) - alphanumeric unique identifier of the title.
ordering (integer) – a number to uniquely identify rows for a given titleId.
nconst (string) - alphanumeric unique identifier of the name/person.
category (string) - the category of job that person was in.
job (string) - the specific job title if applicable, else.
characters (string) - the name of the character played if applicable, else.

**title.ratings.tsv.gz – Contains the IMDb rating and votes information for titles:**
tconst (string) - alphanumeric unique identifier of the title.
averageRating – weighted average of all the individual user ratings.
numVotes - number of votes the title has received.

# PART 1 - Installation and Initial Setup

Begin by installing the necessary libraries that you may need to conduct your analysis. At the very least, you must install pandas and matplotlib

In [3]:
```
%%info
```

Current session configs: `{'proxyUser': 'user_EMR-User', 'conf': {'spark.pyspark.python': 'python3', 'spark.pyspark.virtualenv.enabled': 'true', 'spark.pyspark.virtualenv.type': 'native', 'spark.pyspark.virtualenv.bin.path': '/usr/bin/virtualenv'}, 'kind': 'pyspark'}`

| ID | YARN Application ID | Kind | State | Spark UI | Driver log | User | Current session? |
|---|---|---|---|---|---|---|---|
| 2 | application_1733279512595_0003 | pyspark | idle | Link | Link | None | ✓ |

In [27]:

Current session configs: {'proxyUser': 'user_EMR-User', 'conf': {'spark.pyspark.python': 'python3', 'spark.pyspark.virtualenv.enabled': 'true', 'spark.pyspark.virtualenv.type': 'native', 'spark.pyspark.virtualenv.bin.path': '/usr/bin/virtualenv'}, 'kind': 'pyspark'}

| ID | YARN Application ID | Kind | State | Spark UI | Driver log | User | Current session? |
|---|---|---|---|---|---|---|---|
| 0 | application_1730922819525_0001 | pyspark | idle | Link | Link | None | |
| 1 | application_1730922819525_0002 | pyspark | idle | Link | Link | None | ✓ |

Let's install the necessary packages here

In [4]:
```
sc.install_pypi_package("pandas==1.0.5")
sc.install_pypi_package("matplotlib==3.2.1")
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
Collecting pandas==1.0.5
  Downloading pandas-1.0.5-cp37-cp37m-manylinux1_x86_64.whl (10.1 MB)
Collecting python-dateutil>=2.6.1
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/site-packages (from pandas==1.0.5) (2023.3)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib64/python3.7/site-packages (from pandas==1.0.5) (1.20.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil>=2.6.1->pandas==1.0.5) (1.13.0)
Installing collected packages: python-dateutil, pandas
Successfully installed pandas-1.0.5 python-dateutil-2.9.0.post0

Collecting matplotlib==3.2.1
  Downloading matplotlib-3.2.1-cp37-cp37m-manylinux1_x86_64.whl (12.4 MB)
Collecting cycler>=0.10
  Downloading cycler-0.11.0-py3-none-any.whl (6.4 kB)
Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1
  Downloading pyparsing-3.1.4-py3-none-any.whl (104 kB)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib64/python3.7/site-packages (from matplotlib==3.2.1) (1.20.0)
Requirement already satisfied: python-dateutil>=2.1 in ./tmp/spark-85005380-b8f9-4491-81dd-f675da993490/lib/python3.7/site-packages (from matplotlib==3.2.1) (2.9.0.post0)
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.4.5-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.whl (1.1 MB)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil>=2.1->matplotlib==3.2.1) (1.13.0)
Collecting typing-extensions; python_version < "3.8"
  Downloading typing_extensions-4.7.1-py3-none-any.whl (33 kB)
Installing collected packages: cycler, pyparsing, typing-extensions, kiwisolver, matplotlib
Successfully installed cycler-0.11.0 kiwisolver-1.4.5 matplotlib-3.2.1 pyparsing-3.1.4 typing-extensions-4.7.1

WARNING: The directory '/home/.cache/pip' or its parent directory is not owned or is not writable by the current user. The cache has been disabled. Check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.

WARNING: The directory '/home/.cache/pip' or its parent directory is not owned or is not writable by the current user. The cache has been disabled. Check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.
```

In [3]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
Collecting pandas==1.0.5
  Downloading pandas-1.0.5-cp37-cp37m-manylinux1_x86_64.whl (10.1 MB)
Collecting python-dateutil>=2.6.1
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/site-packages (from pandas==1.0.5) (2023.3)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib64/python3.7/site-packages (from pandas==1.0.5) (1.20.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil>=2.6.1->pandas==1.0.5) (1.
13.0)
Installing collected packages: python-dateutil, pandas
Successfully installed pandas-1.0.5 python-dateutil-2.9.0.post0

Collecting matplotlib==3.2.1
  Downloading matplotlib-3.2.1-cp37-cp37m-manylinux1_x86_64.whl (12.4 MB)
Collecting cycler>=0.10
  Downloading cycler-0.11.0-py3-none-any.whl (6.4 kB)
Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1
  Downloading pyparsing-3.1.4-py3-none-any.whl (104 kB)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib64/python3.7/site-packages (from matplotlib==3.2.1) (1.20.0)
Requirement already satisfied: python-dateutil>=2.1 in ./tmp/spark-bc6b2510-a0a5-4d1b-83c3-d6764d7c491a/lib/python3.7/site-package
s (from matplotlib==3.2.1) (2.9.0.post0)
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.4.5-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.whl (1.1 MB)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/site-packages (from python-dateutil>=2.1->matplotlib==3.2.1)
(1.13.0)
Collecting typing-extensions; python_version < "3.8"
  Downloading typing_extensions-4.7.1-py3-none-any.whl (33 kB)
Installing collected packages: cycler, pyparsing, typing-extensions, kiwisolver, matplotlib
Successfully installed cycler-0.11.0 kiwisolver-1.4.5 matplotlib-3.2.1 pyparsing-3.1.4 typing-extensions-4.7.1

WARNING: The directory '/home/.cache/pip' or its parent directory is not owned or is not writable by the current user. The cache h
as been disabled. Check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.

WARNING: The directory '/home/.cache/pip' or its parent directory is not owned or is not writable by the current user. The cache h
as been disabled. Check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.
```

Now, import the installed packages from the previous block below.

In [5]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [4]:
```
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

# Loading Data

Load all data from S3 into a Spark dataframe object

In [6]:
```
name = spark.read.csv('s3://cis9760-lecture9-movieanalysis/name.basics.new.tsv', sep=r'\t', header=True)
titles = spark.read.csv('s3://cis9760-lecture9-movieanalysis/title.basic.new.tsv', sep=r'\t', header=True)
principles = spark.read.csv('s3://cis9760-lecture9-movieanalysis/title.principles.new.tsv', sep=r'\t', header=True)
ratings = spark.read.csv('s3://cis9760-lecture9-movieanalysis/title.ratings.new.tsv', sep=r'\t', header=True)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

In [49]:
```
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

## Name Basics

Display the schema below:

In [7]:
```
name.printSchema()
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
root
 |-- nconst: string (nullable = true)
 |-- primaryName: string (nullable = true)
 |-- birthYear: string (nullable = true)
 |-- deathYear: string (nullable = true)
 |-- primaryProfession: string (nullable = true)
 |-- knownForTitles: string (nullable = true)
```

In [50]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
root
 |-- nconst: string (nullable = true)
 |-- primaryName: string (nullable = true)
 |-- birthYear: string (nullable = true)
 |-- deathYear: string (nullable = true)
 |-- primaryProfession: string (nullable = true)
 |-- knownForTitles: string (nullable = true)
```

Display the first 15 rows with the following columns:

- `nconst`
- `primaryName`
- `primaryProfession`
- `birthYear`

In [8]:

```
name.select("nconst", "primaryName", "primaryProfession", "birthYear").show(15, truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+---------+------------------+-----------------------------------+---------+
|nconst   |primaryName       |primaryProfession                  |birthYear|
+---------+------------------+-----------------------------------+---------+
|nm0000001|Fred Astaire      |soundtrack,actor,miscellaneous     |1899     |
|nm0000002|Lauren Bacall     |actress,soundtrack                 |1924     |
|nm0000003|Brigitte Bardot   |actress,soundtrack,music_department|1934     |
|nm0000004|John Belushi      |actor,soundtrack,writer            |1949     |
|nm0000005|Ingmar Bergman    |writer,director,actor              |1918     |
|nm0000006|Ingrid Bergman    |actress,soundtrack,producer        |1915     |
|nm0000007|Humphrey Bogart   |actor,soundtrack,producer          |1899     |
|nm0000008|Marlon Brando     |actor,soundtrack,director          |1924     |
|nm0000009|Richard Burton    |actor,soundtrack,producer          |1925     |
|nm0000010|James Cagney      |actor,soundtrack,director          |1899     |
|nm0000011|Gary Cooper       |actor,soundtrack,stunts            |1901     |
|nm0000012|Bette Davis       |actress,soundtrack,make_up_department|1908     |
|nm0000013|Doris Day         |soundtrack,actress,producer        |1922     |
|nm0000014|Olivia de Havilland|actress,soundtrack                 |1916     |
|nm0000015|James Dean        |actor,miscellaneous                |1931     |
+---------+------------------+-----------------------------------+---------+
only showing top 15 rows
```

In [51]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+---------+------------------+-----------------------------------+---------+
|nconst   |primaryName       |primaryProfession                  |birthYear|
+---------+------------------+-----------------------------------+---------+
|nm0000001|Fred Astaire      |soundtrack,actor,miscellaneous     |1899     |
|nm0000002|Lauren Bacall     |actress,soundtrack                 |1924     |
|nm0000003|Brigitte Bardot   |actress,soundtrack,music_department|1934     |
|nm0000004|John Belushi      |actor,soundtrack,writer            |1949     |
|nm0000005|Ingmar Bergman    |writer,director,actor              |1918     |
|nm0000006|Ingrid Bergman    |actress,soundtrack,producer        |1915     |
|nm0000007|Humphrey Bogart   |actor,soundtrack,producer          |1899     |
|nm0000008|Marlon Brando     |actor,soundtrack,director          |1924     |
|nm0000009|Richard Burton    |actor,soundtrack,producer          |1925     |
|nm0000010|James Cagney      |actor,soundtrack,director          |1899     |
|nm0000011|Gary Cooper       |actor,soundtrack,stunts            |1901     |
|nm0000012|Bette Davis       |actress,soundtrack,make_up_department|1908     |
|nm0000013|Doris Day         |soundtrack,actress,producer        |1922     |
|nm0000014|Olivia de Havilland|actress,soundtrack                 |1916     |
|nm0000015|James Dean        |actor,miscellaneous                |1931     |
+---------+------------------+-----------------------------------+---------+
only showing top 15 rows
```

## Title Basics

Display the first 5 rows with the following columns:

- `tconst`
- `titleType`
- `primaryTitle`
- `genres`

In [9]:

```
titles.select("tconst", "titleType", "primaryTitle", "genres").show(5, truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+---------+---------+---------------------+-----------------------+
|tconst   |titleType|primaryTitle         |genres                 |
+---------+---------+---------------------+-----------------------+
|tt0000001|short    |Carmencita           |Documentary,Short      |
|tt0000002|short    |Le clown et ses chiens|Animation,Short        |
|tt0000003|short    |Pauvre Pierrot       |Animation,Comedy,Romance|
|tt0000004|short    |Un bon bock          |Animation,Short        |
|tt0000005|short    |Blacksmith Scene     |Comedy,Short           |
+---------+---------+---------------------+-----------------------+
only showing top 5 rows
```

In [52]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+---------+---------+---------------------+-----------------------+
|tconst   |titleType|primaryTitle         |genres                 |
+---------+---------+---------------------+-----------------------+
|tt0000001|short    |Carmencita           |Documentary,Short      |
|tt0000002|short    |Le clown et ses chiens|Animation,Short        |
|tt0000003|short    |Pauvre Pierrot       |Animation,Comedy,Romance|
|tt0000004|short    |Un bon bock          |Animation,Short        |
|tt0000005|short    |Blacksmith Scene     |Comedy,Short           |
+---------+---------+---------------------+-----------------------+
only showing top 5 rows
```

Display the unique title types below:

In [10]:

```
titles.select("titleType").distinct().show(truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+------------+
|titleType   |
+------------+
|tvSeries    |
|tvMiniSeries|
|movie       |
|videoGame   |
|tvSpecial   |
|video       |
|tvMovie     |
|tvEpisode   |
|tvShort     |
|short       |
|tvPilot     |
+------------+
```

In [53]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+------------+
|titleType   |
+------------+
|tvSeries    |
|tvMiniSeries|
|movie       |
|videoGame   |
|tvSpecial   |
|video       |
|tvMovie     |
|tvEpisode   |
|tvShort     |
|short       |
|tvPilot     |
+------------+
```

Display the schema below:

In [11]:

```
titles.printSchema()
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
root
 |-- tconst: string (nullable = true)
 |-- titleType: string (nullable = true)
 |-- primaryTitle: string (nullable = true)
 |-- originalTitle: string (nullable = true)
 |-- isAdult: string (nullable = true)
 |-- startYear: string (nullable = true)
 |-- endYear: string (nullable = true)
 |-- runtimeMinutes: string (nullable = true)
 |-- genres: string (nullable = true)
```

In [54]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
root
 |-- tconst: string (nullable = true)
 |-- titleType: string (nullable = true)
 |-- primaryTitle: string (nullable = true)
 |-- originalTitle: string (nullable = true)
 |-- isAdult: string (nullable = true)
 |-- startYear: string (nullable = true)
 |-- endYear: string (nullable = true)
 |-- runtimeMinutes: string (nullable = true)
 |-- genres: string (nullable = true)
```

Remove the 'originalTitle' from the dataframe and display the schema to verify it.

In [12]:

```
titles=titles.drop("originalTitle")
titles.printSchema()
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
root
 |-- tconst: string (nullable = true)
 |-- titleType: string (nullable = true)
 |-- primaryTitle: string (nullable = true)
 |-- isAdult: string (nullable = true)
 |-- startYear: string (nullable = true)
 |-- endYear: string (nullable = true)
 |-- runtimeMinutes: string (nullable = true)
 |-- genres: string (nullable = true)
```

In [55]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
root
 |-- tconst: string (nullable = true)
 |-- titleType: string (nullable = true)
 |-- primaryTitle: string (nullable = true)
 |-- isAdult: string (nullable = true)
 |-- startYear: string (nullable = true)
 |-- endYear: string (nullable = true)
 |-- runtimeMinutes: string (nullable = true)
 |-- genres: string (nullable = true)
```

## Title Principles

Display the schema below:

In [13]:

```
principles.printSchema()
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
root
 |-- tconst: string (nullable = true)
 |-- ordering: string (nullable = true)
 |-- nconst: string (nullable = true)
 |-- category: string (nullable = true)
 |-- job: string (nullable = true)
 |-- characters: string (nullable = true)
```

In [56]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
root
 |-- tconst: string (nullable = true)
 |-- ordering: string (nullable = true)
 |-- nconst: string (nullable = true)
 |-- category: string (nullable = true)
 |-- job: string (nullable = true)
 |-- characters: string (nullable = true)
```

Display the first 15 rows where the "category" column is "producer"

In [14]:
```
from pyspark.sql.functions import col
principles.filter(col("category") == 'producer').show(15, truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+---------+--------+---------+--------+--------+----------+
|tconst   |ordering|nconst   |category|job     |characters|
+---------+--------+---------+--------+--------+----------+
|tt0000003|2       |nm1770680|producer|producer|\N        |
|tt0000005|4       |nm0249379|producer|producer|\N        |
|tt0000007|5       |nm0249379|producer|producer|\N        |
|tt0000020|2       |nm0666972|producer|producer|\N        |
|tt0000024|4       |nm0666972|producer|producer|\N        |
|tt0000025|2       |nm0666972|producer|producer|\N        |
|tt0000039|1       |nm0666972|producer|producer|\N        |
|tt0000041|2       |nm0525908|producer|producer|\N        |
|tt0000061|3       |nm0666972|producer|producer|\N        |
|tt0000089|3       |nm0525910|producer|producer|\N        |
|tt0000104|1       |nm0525910|producer|producer|\N        |
|tt0000121|5       |nm0666972|producer|producer|\N        |
|tt0000125|1       |nm0666972|producer|producer|\N        |
|tt0000147|6       |nm0103755|producer|producer|\N        |
|tt0000160|2       |nm0666972|producer|producer|\N        |
+---------+--------+---------+--------+--------+----------+
only showing top 15 rows
```

In [57]:
```

```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+---------+--------+---------+--------+--------+----------+
|tconst   |ordering|nconst   |category|job     |characters|
+---------+--------+---------+--------+--------+----------+
|tt0000003|2       |nm1770680|producer|producer|\N        |
|tt0000005|4       |nm0249379|producer|producer|\N        |
|tt0000007|5       |nm0249379|producer|producer|\N        |
|tt0000020|2       |nm0666972|producer|producer|\N        |
|tt0000024|4       |nm0666972|producer|producer|\N        |
|tt0000025|2       |nm0666972|producer|producer|\N        |
|tt0000039|1       |nm0666972|producer|producer|\N        |
|tt0000041|2       |nm0525908|producer|producer|\N        |
|tt0000061|3       |nm0666972|producer|producer|\N        |
|tt0000089|3       |nm0525910|producer|producer|\N        |
|tt0000104|1       |nm0525910|producer|producer|\N        |
|tt0000121|5       |nm0666972|producer|producer|\N        |
|tt0000125|1       |nm0666972|producer|producer|\N        |
|tt0000147|6       |nm0103755|producer|producer|\N        |
|tt0000160|2       |nm0666972|producer|producer|\N        |
+---------+--------+---------+--------+--------+----------+
only showing top 15 rows
```

## Title Ratings

Display the schema below:

In [15]:
```
ratings.printSchema()
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
root
 |-- tconst: string (nullable = true)
 |-- averageRating: string (nullable = true)
 |-- numVotes: string (nullable = true)
```

In [58]:
```

```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
root
 |-- tconst: string (nullable = true)
 |-- averageRating: string (nullable = true)
 |-- numVotes: string (nullable = true)
```

Display the first 10 rows in a descending order by the number of votes

In [16]:
```
ratings=ratings.withColumn("numVotes", col("numVotes").cast("int"))
ratings.sort(ratings.numVotes.desc()).show(10)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+---------+-------------+--------+
|   tconst|averageRating|numVotes|
+---------+-------------+--------+
|tt0111161|          9.3| 2868594|
|tt0468569|          9.0| 2850372|
|tt1375666|          8.8| 2531543|
|tt0137523|          8.8| 2303989|
|tt0944947|          9.2| 2265760|
|tt0109830|          8.8| 2239746|
|tt0110912|          8.9| 2203191|
|tt0903747|          9.5| 2114358|
|tt0816692|          8.7| 2073181|
|tt0133093|          8.7| 2038364|
+---------+-------------+--------+
only showing top 10 rows
```

In [59]:
```

```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+---------+-------------+--------+
|   tconst|averageRating|numVotes|
+---------+-------------+--------+
|tt0111161|          9.3| 2868594|
|tt0468569|          9.0| 2850372|
|tt1375666|          8.8| 2531543|
|tt0137523|          8.8| 2303989|
|tt0944947|          9.2| 2265760|
|tt0109830|          8.8| 2239746|
|tt0110912|          8.9| 2203191|
|tt0903747|          9.5| 2114358|
|tt0816692|          8.7| 2073181|
|tt0133093|          8.7| 2038364|
+---------+-------------+--------+
only showing top 10 rows
```

# Overview of Data

Display the number of rows and columns in each dataFrame object.

In [17]:
```
print("Number of columns in Name Basics table:", len(name.dtypes))
print("Number of rows in Name Basics table:", name.count())

print("Number of columns in Title Basics table:", len(titles.dtypes))
print("Number of rows in Title Basics table:", titles.count())

print("Number of columns in Title Principles table:", len(principles.dtypes))
print("Number of rows in Title Principles table:", principles.count())

print("Number of columns in Title Ratings table:", len(ratings.dtypes))
print("Number of rows in Title Ratings table:", ratings.count())
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
Number of columns in Name Basics table: 6
Number of rows in Name Basics table: 13329316
Number of columns in Title Basics table: 8
Number of rows in Title Basics table: 10613322
Number of columns in Title Principles table: 6
Number of rows in Title Principles table: 60833800
Number of columns in Title Ratings table: 3
Number of rows in Title Ratings table: 1412275
```

In [60]:
```

```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
Number of columns in Name Basics table: 6
Number of rows in Name Basics table: 13329316

Number of columns in Title Basics table: 8
Number of rows in Title Basics table: 10613322

Number of columns in Title Principles table: 6
Number of rows in Title Principles table: 60833800

Number of columns in Title Ratings table: 3
Number of rows in Title Ratings table: 1412275
```

# PART 2 - Analyzing Movie Genres

Let's now answer this question: how many unique movie genres are represented in this dataset?

Essentially, we have the genres per movie as a list - this is useful to quickly see what each movie might be represented as but it is difficult to easily answer questions such as:

- How many movies are categorized as Comedy, for instance?
- What are the top 20 most popular genres available?

## Association Table

We need to "break out" these genres from the tconst? One common approach to take is to build an association table mapping a single tconst multiple times to each distinct genre.

For instance, given the following:

| tconst | titleType | genres |
|---|---|---|
| abcd123 | XXX | a,b,c |

We would like to derive something like:

| tconst | titleType | genre |
|---|---|---|
| abcd123 | XXX | a |
| abcd123 | XXX | b |
| abcd123 | XXX | c |

What this does is allow us to then perform a myriad of rollups and other analysis on this association table which can aid us in answering the questions asked above.

Implement the code necessary to derive the table described from the data set

In [18]:
```
titles.select("tconst", "titleType", "genres").show(15, truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+---------+---------+-----------------------+
|tconst   |titleType|genres                 |
+---------+---------+-----------------------+
|tt0000001|short    |Documentary,Short      |
|tt0000002|short    |Animation,Short        |
|tt0000003|short    |Animation,Comedy,Romance|
|tt0000004|short    |Animation,Short        |
|tt0000005|short    |Comedy,Short           |
|tt0000006|short    |Short                  |
|tt0000007|short    |Short,Sport            |
|tt0000008|short    |Documentary,Short      |
|tt0000009|movie    |Romance                |
|tt0000010|short    |Documentary,Short      |
|tt0000011|short    |Documentary,Short      |
|tt0000012|short    |Documentary,Short      |
|tt0000013|short    |Documentary,Short      |
|tt0000014|short    |Comedy,Short           |
|tt0000015|short    |Animation,Short        |
+---------+---------+-----------------------+
only showing top 15 rows
```

In [61]:
```

```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
+---------+---------+-----------------------+
|tconst   |titleType|genres                 |
+---------+---------+-----------------------+
|tt0000001|short    |Documentary,Short      |
|tt0000002|short    |Animation,Short        |
|tt0000003|short    |Animation,Comedy,Romance|
|tt0000004|short    |Animation,Short        |
|tt0000005|short    |Comedy,Short           |
|tt0000006|short    |Short                  |
|tt0000007|short    |Short,Sport            |
|tt0000008|short    |Documentary,Short      |
|tt0000009|movie    |Romance                |
|tt0000010|short    |Documentary,Short      |
|tt0000011|short    |Documentary,Short      |
|tt0000012|short    |Documentary,Short      |
|tt0000013|short    |Documentary,Short      |
|tt0000014|short    |Comedy,Short           |
|tt0000015|short    |Animation,Short        |
+---------+---------+-----------------------+
only showing top 15 rows
```

Display the first 25 rows of your association table below

In [19]:
```
from pyspark.sql.functions import split, col, explode
genre=titles.select("tconst", "titleType", "genres")\
    .withColumn("genres",explode(split("genres",",")))\
    .withColumnRenamed('genres', 'Genre')
genre.show(25, truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+---------+---------+-----------+
|tconst   |titleType|Genre      |
+---------+---------+-----------+
|tt0000001|short    |Documentary|
|tt0000001|short    |Short      |
|tt0000002|short    |Animation  |
|tt0000002|short    |Short      |
|tt0000003|short    |Animation  |
|tt0000003|short    |Comedy     |
|tt0000003|short    |Romance    |
|tt0000004|short    |Animation  |
|tt0000004|short    |Short      |
|tt0000005|short    |Comedy     |
|tt0000005|short    |Short      |
|tt0000006|short    |Short      |
|tt0000007|short    |Short      |
|tt0000007|short    |Sport      |
|tt0000008|short    |Documentary|
|tt0000008|short    |Short      |
|tt0000009|movie    |Romance    |
|tt0000010|short    |Documentary|
|tt0000010|short    |Short      |
|tt0000011|short    |Documentary|
|tt0000011|short    |Short      |
|tt0000012|short    |Documentary|
|tt0000012|short    |Short      |
|tt0000013|short    |Documentary|
|tt0000013|short    |Short      |
+---------+---------+-----------+
only showing top 25 rows
```

In [62]:
```

```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
+---------+---------+-----------+
|tconst   |titleType|Genre      |
+---------+---------+-----------+
|tt0000001|short    |Documentary|
|tt0000001|short    |Short      |
|tt0000002|short    |Animation  |
|tt0000002|short    |Short      |
|tt0000003|short    |Animation  |
|tt0000003|short    |Comedy     |
|tt0000003|short    |Romance    |
|tt0000004|short    |Animation  |
|tt0000004|short    |Short      |
|tt0000005|short    |Comedy     |
|tt0000005|short    |Short      |
|tt0000006|short    |Short      |
|tt0000007|short    |Short      |
|tt0000007|short    |Sport      |
|tt0000008|short    |Documentary|
|tt0000008|short    |Short      |
|tt0000009|movie    |Romance    |
|tt0000010|short    |Documentary|
|tt0000010|short    |Short      |
|tt0000011|short    |Documentary|
|tt0000011|short    |Short      |
|tt0000012|short    |Documentary|
|tt0000012|short    |Short      |
|tt0000013|short    |Documentary|
|tt0000013|short    |Short      |
+---------+---------+-----------+
only showing top 25 rows
```

## Total Unique Movie Genres

What is the total number of unique movie genres?

In [20]:
```
from pyspark.sql.functions import countDistinct
genre.filter(genre.titleType == "movie").select(countDistinct("Genre")).collect()[0][0]
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
29
```

In [63]:
```

```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
29
```

What are the unique movie genres?

In [21]:
```
genre.filter(genre.titleType == "movie").select("Genre").distinct().show(29, truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
+-----------+
|Genre      |
+-----------+
|Mystery    |
|Musical    |
|Sport      |
|Action     |
|Talk-Show  |
|Romance    |
|Thriller   |
|\N         |
|Reality-TV |
|Family     |
|Fantasy    |
|History    |
|Animation  |
|Film-Noir  |
|Short      |
|Sci-Fi     |
|News       |
|Drama      |
|Documentary|
|Western    |
|Comedy     |
|Crime      |
|War        |
|Game-Show  |
|Adult      |
|Music      |
|Biography  |
|Adventure  |
|Horror     |
+-----------+
```

In [64]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+-----------+
|genre      |
+-----------+
|Mystery    |
|Musical    |
|Sport      |
|Action     |
|Talk-Show  |
|Romance    |
|Thriller   |
|\N         |
|Reality-TV |
|Family     |
|Fantasy    |
|History    |
|Animation  |
|Film-Noir  |
|Short      |
|Sci-Fi     |
|News       |
|Drama      |
|Documentary|
|Western    |
|Comedy     |
|Crime      |
|War        |
|Game-Show  |
|Adult      |
|Music      |
|Biography  |
|Adventure  |
|Horror     |
+-----------+
```

Oops! Something is off!

In [22]:

```
from pyspark.sql.functions import col
nll = '\\N'
genre.filter(genre.titleType == "movie")\
    .select("Genre").distinct().filter(col("Genre") != nll).show(29, truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
+-----------+
|Genre      |
+-----------+
|Mystery    |
|Musical    |
|Sport      |
|Action     |
|Talk-Show  |
|Romance    |
|Thriller   |
|Reality-TV |
|Family     |
|Fantasy    |
|History    |
|Animation  |
|Film-Noir  |
|Short      |
|Sci-Fi     |
|News       |
|Drama      |
|Documentary|
|Western    |
|Comedy     |
|Crime      |
|War        |
|Game-Show  |
|Adult      |
|Music      |
|Biography  |
|Adventure  |
|Horror     |
+-----------+
```

In [65]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+-----------+
|genre      |
+-----------+
|Mystery    |
|Musical    |
|Sport      |
|Action     |
|Talk-Show  |
|Romance    |
|Thriller   |
|Reality-TV |
|Family     |
|Fantasy    |
|History    |
|Animation  |
|Film-Noir  |
|Short      |
|Sci-Fi     |
|News       |
|Drama      |
|Documentary|
|Western    |
|Comedy     |
|Crime      |
|War        |
|Game-Show  |
|Adult      |
|Music      |
|Biography  |
|Adventure  |
|Horror     |
+-----------+
```

## Top Genres by Movies

Now let's find the highest rated genres in this dataset by rolling up genres.

### Average Rating / Genre

So now, let's unroll our distinct count a bit and display the per average rating value of per genre.

The expected output should be:

| genre | averageRating |
|---|---|
| a | 8.5 |
| b | 6.3 |
| c | 7.2 |

Or something to that effect.

First, let's join our two dataframes (title ratings and title basics) by tconst. Use inner join.

In [23]:
```
ratings=ratings.withColumn("averageRating", col("averageRating").cast("float"))
nll = '\\N'
joined_genre=ratings.join(titles, on='tconst', how='inner')\
    .select("genres", "averageRating")\
    .withColumn('genres',explode(split('genres',",")))\
    .withColumnRenamed('genres', 'Genre')\
    .filter((col("genres") != nll) & (titles.titleType == "movie"))
joined_genre.show(10)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+---------+-------------+
|    Genre|averageRating|
+---------+-------------+
|    Drama|          4.2|
|    Drama|          4.5|
|Biography|          3.6|
|    Drama|          3.6|
|  History|          3.6|
|    Drama|          6.0|
|    Drama|          5.0|
|  History|          5.0|
|Biography|          6.2|
|    Drama|          6.2|
+---------+-------------+
only showing top 10 rows
```

In [66]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+---------+-------------+
|    Genre|averageRating|
+---------+-------------+
|    Drama|          4.2|
|    Drama|          4.5|
|Biography|          3.6|
|    Drama|          3.6|
|  History|          3.6|
|    Drama|          6.0|
|    Drama|          5.0|
|  History|          5.0|
|Biography|          6.2|
|    Drama|          6.2|
+---------+-------------+
only showing top 10 rows
```

Now, let's aggregate along the averageRating column to get a resultant dataframe that displays average rating per genre.

In [24]:
```
from pyspark.sql.functions import mean, round
joined_genre.groupBy("Genre").agg(
        round(mean('averageRating'),3).alias("Rating")).show(truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
+-----------+------+
|Genre      |Rating|
+-----------+------+
|Mystery    |5.847 |
|Musical    |6.187 |
|Action     |5.732 |
|Sport      |6.623 |
|Talk-Show  |6.858 |
|Romance    |6.102 |
|Thriller   |5.613 |
|Reality-TV |6.701 |
|Family     |6.205 |
|Fantasy    |5.898 |
|History    |6.798 |
|Animation  |6.367 |
|Film-Noir  |6.463 |
|Sci-Fi     |5.353 |
|News       |7.203 |
|Drama      |6.248 |
|Documentary|7.216 |
|Western    |5.84  |
|Comedy     |5.906 |
|Crime      |5.985 |
+-----------+------+
only showing top 20 rows
```

In [67]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+-----------+------+
|Genre      |Rating|
+-----------+------+
|Mystery    |5.847 |
|Musical    |6.187 |
|Action     |5.732 |
|Sport      |6.623 |
|Talk-Show  |6.858 |
|Romance    |6.102 |
|Thriller   |5.613 |
|Reality-TV |6.701 |
|Family     |6.205 |
|Fantasy    |5.898 |
|History    |6.798 |
|Animation  |6.367 |
|Film-Noir  |6.463 |
|Sci-Fi     |5.353 |
|News       |7.203 |
|Drama      |6.248 |
|Documentary|7.216 |
|Western    |5.84  |
|Comedy     |5.906 |
|Crime      |5.985 |
+-----------+------+
only showing top 20 rows
```

## Horizontal Bar Chart of Top Genres

With this data available, let us now build a barchart of all genres

**HINT**: don't forget about the matplotlib magic!

```
%matplot plt
```

In [25]:

```
from pyspark.sql.functions import mean, round
genre_rating=joined_genre.groupBy("Genre").agg(
        round(mean('averageRating'),3).alias("Rating")).sort(col("Rating").desc())
genre_rating.show(29, truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
+-----------+------+
|Genre      |Rating|
+-----------+------+
|Documentary|7.216 |
|News       |7.203 |
|Biography  |6.951 |
|Game-Show  |6.88  |
|Talk-Show  |6.858 |
|History    |6.798 |
|Music      |6.755 |
|Reality-TV |6.701 |
|Sport      |6.623 |
|Film-Noir  |6.463 |
|War        |6.403 |
|Animation  |6.367 |
|Drama      |6.248 |
|Family     |6.205 |
|Musical    |6.187 |
|Romance    |6.102 |
|Crime      |5.985 |
|Comedy     |5.906 |
|Fantasy    |5.898 |
|Adventure  |5.866 |
|Mystery    |5.847 |
|Western    |5.84  |
|Action     |5.732 |
|Thriller   |5.613 |
|Adult      |5.554 |
|Sci-Fi     |5.353 |
|Horror     |5.002 |
|Short      |5.0   |
+-----------+------+
```

In [68]:

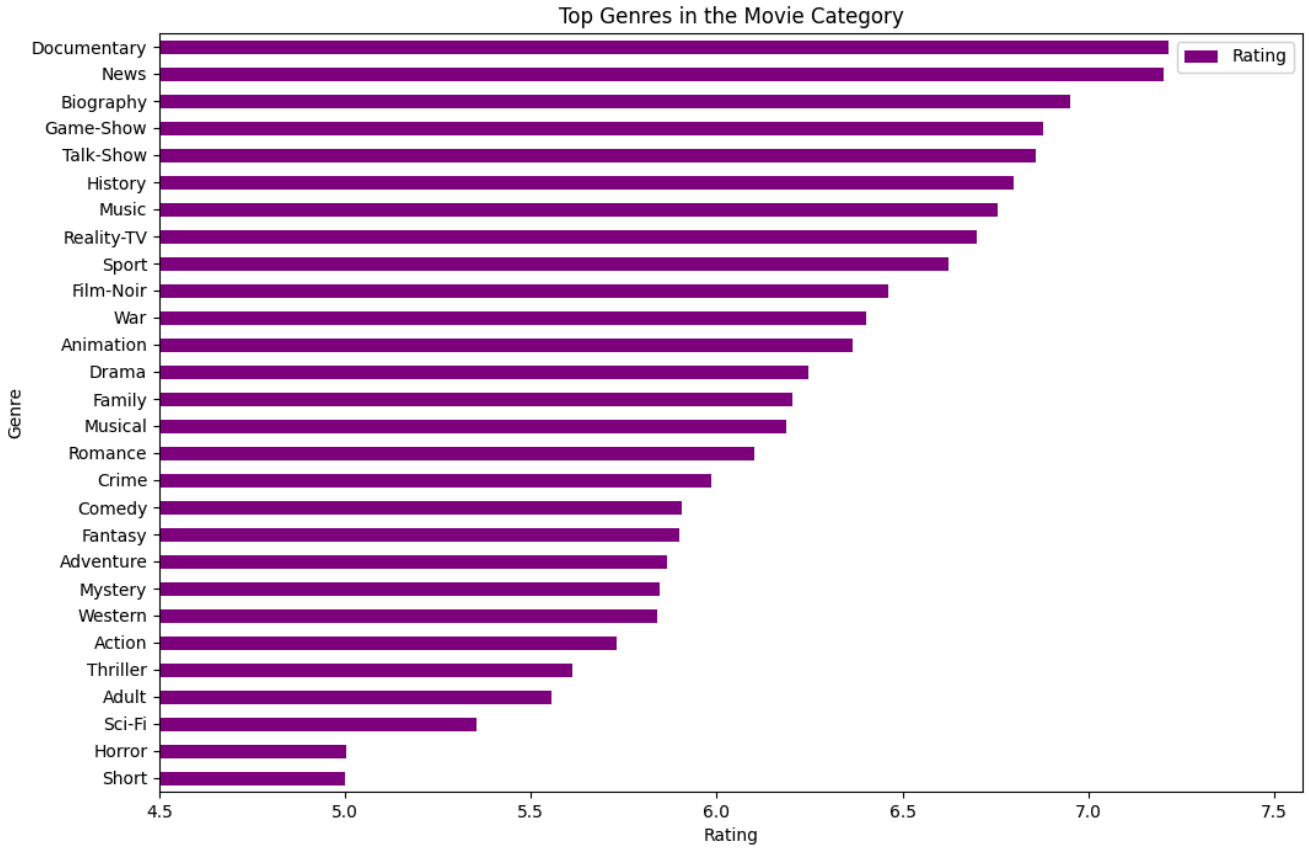
```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+-----------+------+
|Genre      |Rating|
+-----------+------+
|Documentary|7.216 |
|News       |7.203 |
|Biography  |6.951 |
|Game-Show  |6.88  |
|Talk-Show  |6.858 |
|History    |6.798 |
|Music      |6.755 |
|Reality-TV |6.701 |
|Sport      |6.623 |
|Film-Noir  |6.463 |
|War        |6.403 |
|Animation  |6.367 |
|Drama      |6.248 |
|Family     |6.205 |
|Musical    |6.187 |
|Romance    |6.102 |
|Crime      |5.985 |
|Comedy     |5.906 |
|Fantasy    |5.898 |
|Adventure  |5.866 |
|Mystery    |5.847 |
|Western    |5.84  |
|Action     |5.732 |
|Thriller   |5.613 |
|Adult      |5.554 |
|Sci-Fi     |5.353 |
|Horror     |5.002 |
|Short      |5.0   |
+-----------+------+
```

You do not have to match the color and the figure size but all other aspects of the graph should be matched.

In [28]:

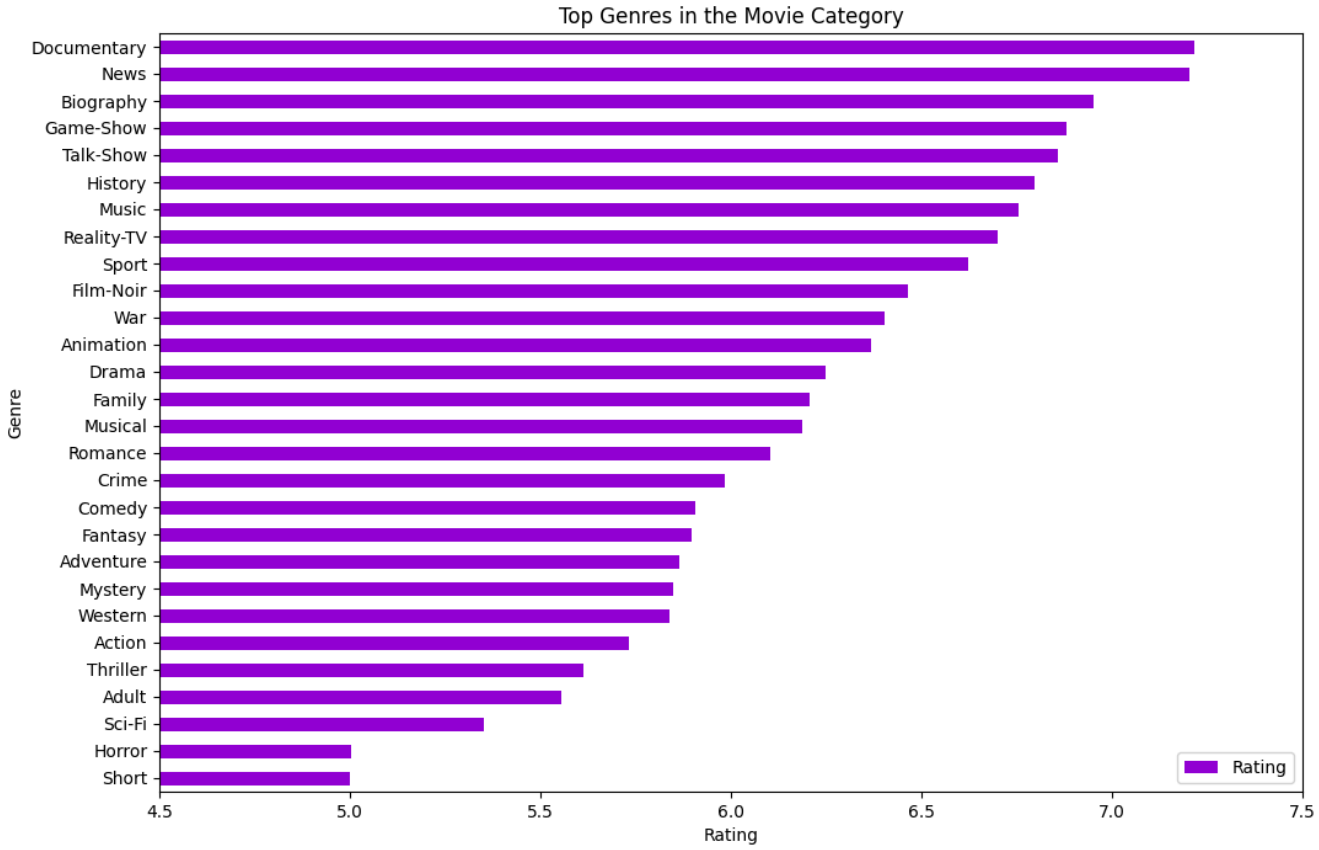
```
genre_rating=genre_rating.sort(col("Rating").asc())
import matplotlib.pyplot as plt
genre_rating.toPandas().plot.barh(x='Genre', y='Rating', color='purple', figsize=(12, 8))
plt.xlim(4.5, None)
plt.xlabel('Rating')
plt.ylabel('Genre')
plt.title('Top Genres in the Movie Category')
%matplot plt
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```




In [70]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```




# PART 3 - Analyzing Job Categories

## Total Unique Job Categories

What is the total number of unique job categories?

In [29]:
```
principles.select('tconst', 'category').show(30, truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
+---------+---------------+
|tconst   |category       |
+---------+---------------+
|tt0000001|self           |
|tt0000001|director       |
|tt0000001|cinematographer|
|tt0000002|director       |
|tt0000002|composer       |
|tt0000003|director       |
|tt0000003|producer       |
|tt0000003|composer       |
|tt0000003|editor         |
|tt0000004|director       |
|tt0000004|composer       |
|tt0000005|actor          |
|tt0000005|actor          |
|tt0000005|director       |
|tt0000005|producer       |
|tt0000006|director       |
|tt0000007|actor          |
|tt0000007|actor          |
|tt0000007|director       |
|tt0000007|director       |
|tt0000007|producer       |
|tt0000008|actor          |
|tt0000008|director       |
|tt0000008|cinematographer|
|tt0000009|actress        |
|tt0000009|actor          |
|tt0000009|actor          |
|tt0000009|director       |
|tt0000010|director       |
|tt0000011|actor          |
+---------+---------------+
only showing top 30 rows
```

In [71]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+---------+---------------+
|tconst   |category       |
+---------+---------------+
|tt0000001|self           |
|tt0000001|director       |
|tt0000001|cinematographer|
|tt0000002|director       |
|tt0000002|composer       |
|tt0000003|director       |
|tt0000003|producer       |
|tt0000003|composer       |
|tt0000003|editor         |
|tt0000004|director       |
|tt0000004|composer       |
|tt0000005|actor          |
|tt0000005|actor          |
|tt0000005|director       |
|tt0000005|producer       |
|tt0000006|director       |
|tt0000007|actor          |
|tt0000007|actor          |
|tt0000007|director       |
|tt0000007|director       |
|tt0000007|producer       |
|tt0000008|actor          |
|tt0000008|director       |
|tt0000008|cinematographer|
|tt0000009|actress        |
|tt0000009|actor          |
|tt0000009|actor          |
|tt0000009|director       |
|tt0000010|director       |
|tt0000011|actor          |
+---------+---------------+
only showing top 30 rows
```

In [30]:

```
from pyspark.sql.functions import countDistinct
principles.select(countDistinct("category")).collect()[0][0]
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
12
```

In [72]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
12
```

What are the unique job categories available?

In [31]:

```
principles.select("category").distinct().show(truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+--------------------+
|category            |
+--------------------+
|actress             |
|producer            |
|production_designer|
|writer              |
|actor               |
|cinematographer     |
|archive_sound       |
|archive_footage     |
|self                |
|editor              |
|composer            |
|director            |
+--------------------+
```

In [73]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+--------------------+
|category            |
+--------------------+
|actress             |
|producer            |
|production_designer|
|writer              |
|actor               |
|cinematographer     |
|archive_sound       |
|archive_footage     |
|self                |
|editor              |
|composer            |
|director            |
+--------------------+
```

# Top Job Categories

Now let's find the top job categories in this dataset by rolling up categories.

## Counts of Titles / Job Category

The expected output should be:

| category | count |
|---|---|
| a | 15 |
| b | 2 |
| c | 45 |

Or something to that effect.

In [32]:

```
from pyspark.sql.functions import count
principles.groupBy("category").agg(count('category').alias("count")).show(truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
+--------------------+--------+
|category            |count   |
+--------------------+--------+
|actress             |10492210|
|producer            |3944711 |
|production_designer|383761  |
|writer              |8495903 |
|actor               |13443688|
|cinematographer     |2068164 |
|archive_sound       |4794    |
|archive_footage     |404581  |
|self                |10562296|
|editor              |2012800 |
|composer            |2014049 |
|director            |7006843 |
+--------------------+--------+
```

In [74]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+--------------------+--------+
|category            |count   |
+--------------------+--------+
|actress             |10492210|
|producer            |3944711 |
|production_designer|383761  |
|writer              |8495903 |
|actor               |13443688|
|cinematographer     |2068164 |
|archive_sound       |4794    |
|archive_footage     |404581  |
|self                |10562296|
|editor              |2012800 |
|composer            |2014049 |
|director            |7006843 |
+--------------------+--------+
```

## Bar Chart of Top Job Categories

With this data available, let us now build a barchart of the top 5 categories.

**HINT**: don't forget about the matplotlib magic!

```
%matplot plt
```

In [33]:

```
from pyspark.sql.functions import count, col
count_job=principles.groupBy("category").agg(count('category').alias("count")).sort(col("count").desc())
count_job.show(truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+--------------------+--------+
|category            |count   |
+--------------------+--------+
|actor               |13443688|
|self                |10562296|
|actress             |10492210|
|writer              |8495903 |
|director            |7006843 |
|producer            |3944711 |
|cinematographer     |2068164 |
|composer            |2014049 |
|editor              |2012800 |
|archive_footage     |404581  |
|production_designer|383761  |
|archive_sound       |4794    |
+--------------------+--------+
```

In [75]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

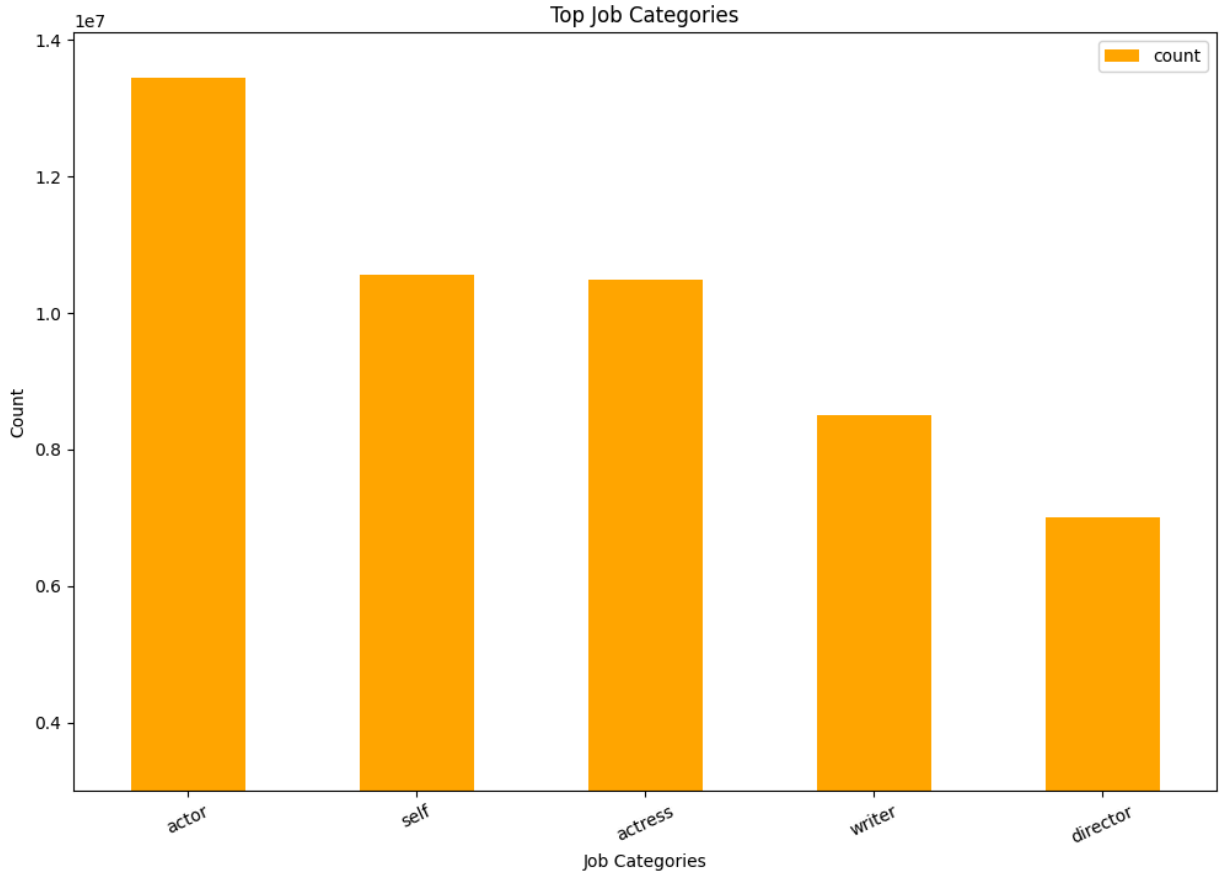
```
+--------------------+--------+
|category            |count   |
+--------------------+--------+
|actor               |13443688|
|self                |10562296|
|actress             |10492210|
|writer              |8495903 |
|director            |7006843 |
|producer            |3944711 |
|cinematographer     |2068164 |
|composer            |2014049 |
|editor              |2012800 |
|archive_footage     |404581  |
|production_designer|383761  |
|archive_sound       |4794    |
+--------------------+--------+
```

You do not have to match the color and the figure size but all other aspects of the graph should be matched.

In [39]:
```
count_job.createOrReplaceTempView('topjob')
sqldf = spark.sql(
'''
SELECT category, count FROM topjob
ORDER BY count DESC
LIMIT 5
'''
)

import matplotlib.pyplot as plt
sqldf.toPandas().plot.bar(x='category', y='count', color='orange', figsize=(12, 8))
plt.ylim(3000000, None)
plt.xlabel('Job Categories')
plt.ylabel('Count')
plt.title('Top Job Categories')
plt.xticks(rotation=25)
%matplot plt
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

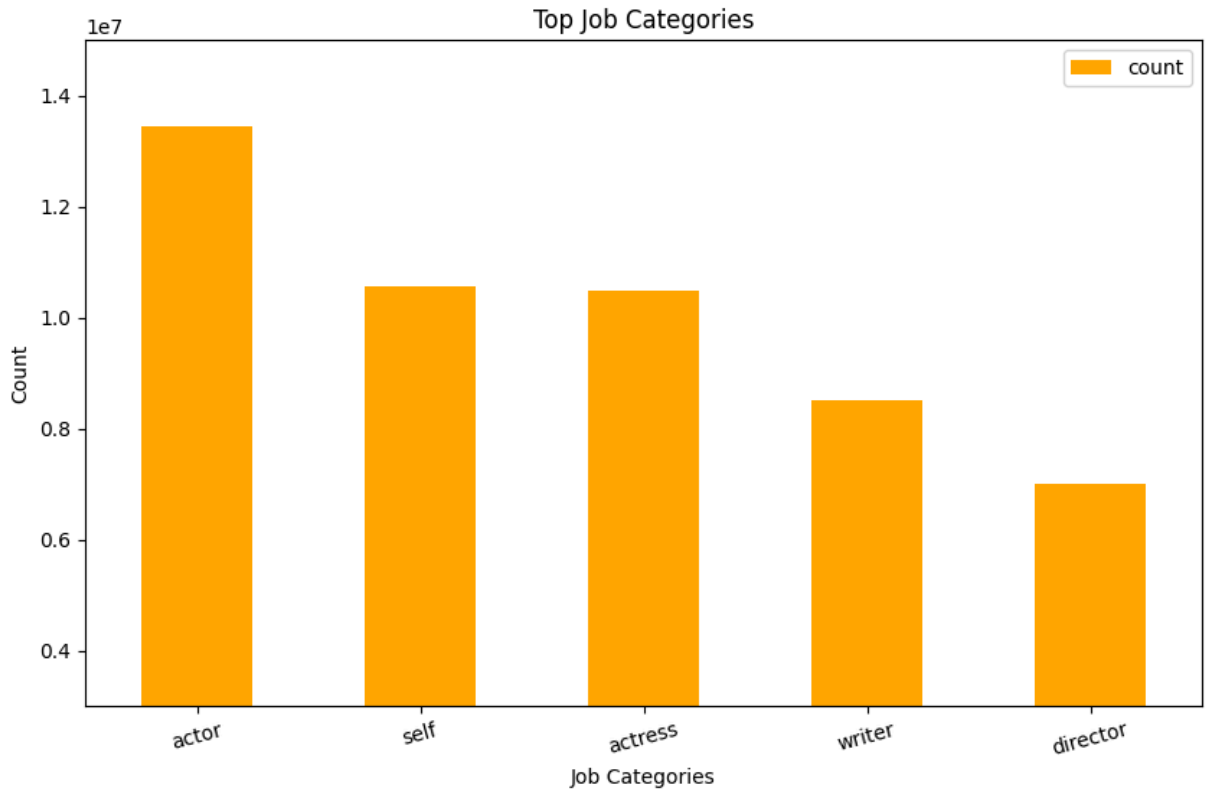



In [76]:
```

```

```
VBox()
```

```
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```




# PART 4 - Answer to the following questions:

1. You will need to join tables to answer the following questions. Not every question will require four tables.
2. Your code should meet all the requirements asked in the questions.
3. Your code should be generalizable enough for any given arguments.

## 1) Which movies, released in 2003, have received more than 50,000 votes and have an average rating of 8 or higher?

In [40]:
```
titles.join(ratings,on='tconst',how='inner')\
    .select("primaryTitle", "averageRating", "numVotes")\
    .withColumnRenamed("primaryTitle","Movie")\
    .withColumnRenamed("averageRating","Ratings")\
    .withColumnRenamed("numVotes","Number of Votes")\
    .filter((titles.titleType == "movie")
            & (titles.startYear == 2003)
            & (ratings.numVotes > 50000)
            & (ratings.averageRating >= 8))\
    .sort(col("averageRating").desc())\
    .show(truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+-----------------------------------------------------+-------+---------------+
|Movie                                                |Ratings|Number of Votes|
+-----------------------------------------------------+-------+---------------+
|The Lord of the Rings: The Return of the King        |9.0    |1965196        |
|Oldboy                                               |8.3    |630695         |
|Finding Nemo                                         |8.2    |1106772        |
|Kill Bill: Vol. 1                                    |8.2    |1184605        |
|Memories of Murder                                   |8.1    |213610         |
|Pirates of the Caribbean: The Curse of the Black Pearl|8.1    |1202458        |
|Munna Bhai M.B.B.S.                                  |8.1    |87972          |
|Spring, Summer, Fall, Winter... and Spring           |8.0    |86510          |
|Dogville                                             |8.0    |157921         |
|Big Fish                                             |8.0    |457515         |
+-----------------------------------------------------+-------+---------------+
```

In [77]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+-----------------------------------------------------+-------+---------------+
|Movie                                                |Ratings|Number of Votes|
+-----------------------------------------------------+-------+---------------+
|The Lord of the Rings: The Return of the King        |9.0    |1965196        |
|Oldboy                                               |8.3    |630695         |
|Finding Nemo                                         |8.2    |1106772        |
|Kill Bill: Vol. 1                                    |8.2    |1184605        |
|Memories of Murder                                   |8.1    |213610         |
|Pirates of the Caribbean: The Curse of the Black Pearl|8.1    |1202458        |
|Munna Bhai M.B.B.S.                                  |8.1    |87972          |
|Spring, Summer, Fall, Winter... and Spring           |8.0    |86510          |
|Dogville                                             |8.0    |157921         |
|Big Fish                                             |8.0    |457515         |
+-----------------------------------------------------+-------+---------------+
```

## 2) List the films featuring Cillian Murphy as an actor since 2007, including their ratings. What is his highest-rated movie?

In [43]:
```
# get all ratings
cm_rating=name.join(principles, on='nconst', how='inner')\
    .join(titles, on='tconst', how='inner')\
    .join(ratings, on='tconst', how='inner')\
    .select("primaryTitle", "startYear", "averageRating")\
    .withColumnRenamed("primaryTitle", "Movies")\
    .withColumnRenamed("startYear", "Year")\
    .withColumnRenamed("averageRating", "Avg Rating")\
    .filter((principles.category == "actor")
            & (name.primaryName == "Cillian Murphy")
            & (titles.startYear >= 2007)
            &(titles.titleType == "movie"))\
    .sort(col("Year").desc())
cm_rating.show(truncate=False)

# get the highest rating
highest=cm_rating.sort(col("Avg Rating").desc()).collect()
print("Highest rated movie:", highest[0][0], "with a rating of", highest[0][2])
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+----------------------+----+----------+
|Movies                |Year|Avg Rating|
+----------------------+----+----------+
|Small Things Like These|2024|7.2       |
|Oppenheimer           |2023|8.4       |
|Kensuke's Kingdom     |2023|7.1       |
|A Quiet Place Part II |2020|7.2       |
|Anna                  |2019|6.6       |
|Anthropoid            |2016|7.2       |
|Free Fire             |2016|6.3       |
|In the Heart of the Sea|2015|6.9       |
|Transcendence         |2014|6.2       |
|Aloft                 |2014|5.3       |
|Red Lights            |2012|6.2       |
|Retreat               |2011|5.8       |
|In Time               |2011|6.7       |
|Peacock               |2010|6.2       |
|Perrier's Bounty      |2009|6.3       |
|Waveriders            |2008|6.8       |
|Sunshine              |2007|7.2       |
|Watching the Detectives|2007|6.2       |
+----------------------+----+----------+

Highest rated movie: Oppenheimer with a rating of 8.4
```

In [79]:
```

```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
+----------------------+----+----------+
|Movies                |Year|Avg Rating|
+----------------------+----+----------+
|Small Things Like These|2024|7.2       |
|Oppenheimer           |2023|8.4       |
|Kensuke's Kingdom     |2023|7.1       |
|A Quiet Place Part II |2020|7.2       |
|Anna                  |2019|6.6       |
|Anthropoid            |2016|7.2       |
|Free Fire             |2016|6.3       |
|In the Heart of the Sea|2015|6.9       |
|Transcendence         |2014|6.2       |
|Aloft                 |2014|5.3       |
|Red Lights            |2012|6.2       |
|Retreat               |2011|5.8       |
|In Time               |2011|6.7       |
|Peacock               |2010|6.2       |
|Perrier's Bounty      |2009|6.3       |
|Waveriders            |2008|6.8       |
|Sunshine              |2007|7.2       |
|Watching the Detectives|2007|6.2       |
+----------------------+----+----------+

Highest rated movie: Oppenheimer with a rating of 8.4
```

## 3) How many movies has Zendaya featured as an actress in each year?

In [44]:
```
name.join(principles, on='nconst', how='inner').join(titles, on='tconst', how='inner')\
    .select("startYear")\
    .withColumnRenamed("startYear", "Year")\
    .filter((name.primaryName == "Zendaya")
            &(principles.category == "actress")
            &(titles.titleType == "movie")
            &(titles.startYear != '\\N'))\
    .groupBy("Year").agg(count('Year').alias("Total"))\
    .sort(col("Year").desc())\
    .show(truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+----+-----+
|Year|Total|
+----+-----+
|2024|2    |
|2021|3    |
|2018|2    |
|2017|1    |
+----+-----+
```

In [80]:
```

```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+----+-----+
|Year|Total|
+----+-----+
|2024|2    |
|2021|3    |
|2018|2    |
|2017|1    |
+----+-----+
```

## 4) At what age did Audrey Hepburn, known for her role in the movie 'Breakfast at Tiffany's,' pass away?

In [45]:
```
age=name.join(principles, on='nconst', how='inner').join(titles, on='tconst', how='inner')\
    .filter((name.primaryName == "Audrey Hepburn")
            &(titles.primaryTitle == "Breakfast at Tiffany's"))\
    .withColumn("Age", col("deathYear") - col("birthYear"))\
    .select("Age")\
    .collect()[0][0]
print(f'Audrey Hepburn passed away at the age of {int(age)}.')
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
Audrey Hepburn passed away at the age of 64.
```

In [87]:
```

```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
```

```
Audrey Hepburn passed away at the age of 64.
```

## 5) What is the movie(s) with the highest average rating among those featuring Chris Evans, known for his role in 'Captain America: The First Avenger'?

Write your code in a way that it finds and displays all movies with the highest rating, even if there's more than one.

In [47]:
```
# get the nconst of Chris Evans
ce_nconst=name.join(principles, on='nconst', how='inner').join(titles, on='tconst', how='inner')\
    .filter((name.primaryName == "Chris Evans")
           &(titles.primaryTitle == "Captain America: The First Avenger"))\
    .select("nconst")\
    .collect()[0][0]

# get the movies Chris Evans acted
movies=principles.join(titles, on='tconst', how='inner').join(ratings, on='tconst', how='inner')\
    .select("primaryTitle", "averageRating")\
    .withColumnRenamed("primaryTitle", "Movies")\
    .withColumnRenamed("averageRating", "Highest Avg Rating")\
    .filter((principles.nconst == ce_nconst)
           &(titles.titleType == "movie"))

# only keep the highest rating
from pyspark.sql.functions import max
highest_rate=movies.select(max("Highest Avg Rating")).collect()[0][0]
movies.filter(col("Highest Avg Rating") == highest_rate).show(truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+----------------------+------------------+
|Movies                |Highest Avg Rating|
+----------------------+------------------+
|Avengers: Infinity War|8.4               |
|Avengers: Endgame     |8.4               |
+----------------------+------------------+
```

In [88]:
```
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+----------------------+------------------+
|Movies                |Highest Avg Rating|
+----------------------+------------------+
|Avengers: Infinity War|8.4               |
|Avengers: Endgame     |8.4               |
+----------------------+------------------+
```

## 6) Among the movies in which Clint Eastwood, known for 'The Good, the Bad and the Ugly', and Harrison Ford, known for 'Raiders of the Lost Ark', have acted, who has the higher average rating?

Hint: You will need to calculate the average rating across all movies for each actor.

In [50]:
```
# get nconst of Harrison Ford
hf_nconst=name.join(principles, on='nconst', how='inner').join(titles, on='tconst', how='inner')\
    .filter((name.primaryName == "Harrison Ford")
           &(titles.primaryTitle == "Raiders of the Lost Ark"))\
    .select("nconst")\
    .collect()[0][0]

# get average rating of Harrison Ford
hf_rating=principles.join(titles, on='tconst', how='inner').join(ratings, on='tconst', how='inner')\
    .select("nconst", "averageRating")\
    .filter((principles.nconst == hf_nconst)
            &(titles.titleType == "movie")
            &(principles.category == "actor"))\
    .groupby("nconst").agg(round(mean("averageRating"),2).alias("rating"))\
    .collect()[0][1]

print('The average rating of Harrison Ford is', hf_rating)

# get nconst of Clint Eastwood
ce_nconst=name.join(principles, on='nconst', how='inner').join(titles, on='tconst', how='inner')\
    .filter((name.primaryName == "Clint Eastwood")
           &(titles.primaryTitle == "The Good, the Bad and the Ugly"))\
    .select("nconst")\
    .collect()[0][0]

# get average rating of Clint Eastwood
```

```
ce_rating=principles.join(titles, on='tconst', how='inner').join(ratings, on='tconst', how='inner')\
    .select("nconst", "averageRating")\
    .filter((principles.nconst == ce_nconst)
            &(titles.titleType == "movie")
            &(principles.category == "actor"))\
    .groupby("nconst").agg(round(mean("averageRating"),2).alias("rating"))\
    .collect()[0][1]

print('The average rating of Clint Eastwood is', ce_rating)

print('Clint Eastwood has a higher average rating')
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
The average rating of Harrison Ford is 6.83
The average rating of Clint Eastwood is 6.86
Clint Eastwood has a higher average rating
```

In [89]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
The average rating of Harrison Ford is 6.83
The average rating of Clint Eastwood is 6.86
Clint Eastwood has a higher average rating
```

## 7) What are the movies in which both Johnny Depp and Helena Bonham Carter have acted together?

In [51]:

```
# get the movies that Johnny Depp acted
jd_movies=name.join(principles, on='nconst', how='inner').join(titles, on='tconst', how='inner')\
        .select("primaryTitle")\
        .filter((name.primaryName == "Johnny Depp")
                &(titles.titleType == "movie")
                &(principles.category == "actor"))

# get the movies that Helena Bonham Carter acted
hb_movies=name.join(principles, on='nconst', how='inner').join(titles, on='tconst', how='inner')\
    .select("primaryTitle")\
    .filter((name.primaryName == "Helena Bonham Carter")
            &(titles.titleType == "movie")
            &(principles.category == "actress"))

# get the common movies
jd_movies.join(hb_movies, on='primaryTitle', how='inner')\
    .select("primaryTitle")\
    .withColumnRenamed("primaryTitle", "Common Movies")\
    .show(truncate=False)
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+---------------------------------------------+
|Common Movies                                |
+---------------------------------------------+
|Dark Shadows                                 |
|Sweeney Todd: The Demon Barber of Fleet Street|
|Corpse Bride                                 |
|Charlie and the Chocolate Factory            |
|Alice in Wonderland                          |
|Alice Through the Looking Glass              |
+---------------------------------------------+
```

In [90]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+---------------------------------------------+
|Common Movies                                |
+---------------------------------------------+
|Dark Shadows                                 |
|Sweeney Todd: The Demon Barber of Fleet Street|
|Corpse Bride                                 |
|Charlie and the Chocolate Factory            |
|Alice in Wonderland                          |
|Alice Through the Looking Glass              |
+---------------------------------------------+
```

## 8) Among the TV series featuring David Tennant, who is known for his role in Doctor Who, which rank in the top 5 for viewer engagement? Does Doctor Who make it into the highest-ranked series?

In [53]:
```
# get the nconst of David Tennant, who features Doctor Who
dt_nconst=name.join(principles, on='nconst', how='inner').join(titles, on='tconst', how='inner')\
        .select("nconst")\
        .filter((name.primaryName == "David Tennant")
                &(principles.category == "actor")
                &(titles.primaryTitle == "Doctor Who"))\
        .distinct()\
        .collect()[0][0]

# numVotes is string for now, change it to int
ratings=ratings.withColumn("numVotes", col("numVotes").cast("int"))

# get the top 5 numVotes using dt_nconst
dt_tv=principles.join(titles, on='tconst', how='inner').join(ratings, on='tconst', how='inner')\
    .select("primaryTitle", "numVotes")\
    .withColumnRenamed("primaryTitle", "Tv Series")\
    .withColumnRenamed("numVotes", "Number of Votes")\
    .filter((principles.nconst == dt_nconst)
            &(titles.titleType == "tvSeries")
            &(principles.category == "actor"))\
    .sort(col("numVotes").desc())\
    .limit(5)

dt_tv.show(truncate=False)

print ("Doctor Who is in the top 5 Tv series for viewer engagement with", dt_tv.collect()[0][1], "votes.")
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+-------------+---------------+
|Tv Series    |Number of Votes|
+-------------+---------------+
|Doctor Who   |245190         |
|Jessica Jones|225869         |
|Broadchurch  |126132         |
|Good Omens   |111450         |
|Ahsoka       |107410         |
+-------------+---------------+

Doctor Who is in the top 5 Tv series for viewer engagement with 245190 votes.
```

In [91]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+-------------+---------------+
|Tv Series    |Number of Votes|
+-------------+---------------+
|Doctor Who   |245190         |
|Jessica Jones|225869         |
|Broadchurch  |126132         |
|Good Omens   |111450         |
|Ahsoka       |107410         |
+-------------+---------------+

Doctor Who is in the top 5 TV series for viewer engagement with 245190 votes.
```

## 9) What are the highest and lowest-rated movies in the Harry Potter franchise featuring Daniel Radcliffe, and what are their ratings?

First, get the ratings for each movie in the franchise, and then find the highest and lowest-rated movies.

In [56]:
```
# get all movies and ratings
hp_movies=name.join(principles, on='nconst', how='inner')\
    .join(titles, on='tconst', how='inner')\
    .join(ratings, on='tconst', how='inner')\
    .select("primaryTitle", "averageRating")\
    .filter((name.primaryName == "Daniel Radcliffe")
            &(titles.primaryTitle.contains( "Harry Potter"))
            &(principles.category == "actor")
            &(titles.titleType == "movie"))
hp_movies.show(truncate=False)

# get the highest rating
highest=hp_movies.sort(col("averageRating").desc()).limit(1).collect()
print("Highest Rating in the Harry Potter Franchise:", highest[0][0], "with a rating of", highest[0][1])

# get the lowest rating
lowest=hp_movies.sort(col("averageRating").asc()).limit(1).collect()
print("Lowest Rating in the Harry Potter Franchise:", lowest[0][0], "with a rating of", lowest[0][1])
```

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+----------------------------------------+-------------+
|primaryTitle                            |averageRating|
+----------------------------------------+-------------+
|Harry Potter and the Half-Blood Prince  |7.6          |
|Harry Potter and the Prisoner of Azkaban|7.9          |
|Harry Potter and the Deathly Hallows: Part 2|8.1          |
|Harry Potter and the Deathly Hallows: Part 1|7.7          |
|Harry Potter and the Chamber of Secrets |7.4          |
|Harry Potter and the Goblet of Fire     |7.7          |
|Harry Potter and the Sorcerer's Stone   |7.6          |
|Harry Potter and the Order of the Phoenix|7.5          |
+----------------------------------------+-------------+

Highest Rating in the Harry Potter Franchise: Harry Potter and the Deathly Hallows: Part 2 with a rating of 8.1
Lowest Rating in the Harry Potter Franchise: Harry Potter and the Chamber of Secrets with a rating of 7.4
```

In [93]:

```
VBox()
FloatProgress(value=0.0, bar_style='info', description='Progress:', layout=Layout(height='25px', width='50%'),…
+----------------------------------------+-------------+
|primaryTitle                            |averageRating|
+----------------------------------------+-------------+
|Harry Potter and the Half-Blood Prince  |7.6          |
|Harry Potter and the Prisoner of Azkaban|7.9          |
|Harry Potter and the Deathly Hallows: Part 2|8.1          |
|Harry Potter and the Deathly Hallows: Part 1|7.7          |
|Harry Potter and the Chamber of Secrets |7.4          |
|Harry Potter and the Goblet of Fire     |7.7          |
|Harry Potter and the Sorcerer's Stone   |7.6          |
|Harry Potter and the Order of the Phoenix|7.5          |
+----------------------------------------+-------------+

Highest Rating in the Harry Potter Franchise: Harry Potter and the Deathly Hallows: Part 2  with a rating of 8.1
Lowest Rating in the Harry Potter Franchise:  Harry Potter and the Chamber of Secrets with a rating of 7.4
```