```
from platform import python_version
print('Versão da Linguagem Python Usada Neste Jupyter Notebook:',
python_version())
```

```
Versão da Linguagem Python Usada Neste Jupyter Notebook: 3.9.7
```

```
!pip install -q imdb-sqlite
!pip install -q pycountry
```

```
import re
import time
import sqlite3
import pycountry
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import cm
from sklearn.feature_extraction.text import CountVectorizer
import warnings
warnings.filterwarnings("ignore")
sns.set_theme(style = "whitegrid")
```

```
%%time
!imdb-sqlite
```

```
Wall time: 167 ms
```

```
2022-09-23 09:52:40,048 DB already exists: (imdb.db). Refusing to
modify. Exiting
```

```
# Conecta no banco de dados
conn = sqlite3.connect("imdb.db")
```

```
# Extrai a lista de tabelas
tabelas = pd.read_sql_query("SELECT NAME AS 'Table_Name' FROM
sqlite_master WHERE type = 'table'", conn)
```

```
# Tipo do objeto
type(tabelas)
```

```
pandas.core.frame.DataFrame
```

```
# Visualiza o resultado
tabelas.head()
```

```
  Table_Name
0     people
1     titles
2       akas
3       crew
4   episodes
```

```
# Vamos converter o dataframe em uma lista
tabelas = tabelas["Table_Name"].values.tolist()

# Vamos percorrer a lista de tabelas no banco de dados e extrair o
esquema de cada uma
for tabela in tabelas:
    consulta = "PRAGMA TABLE_INFO({})".format(tabela)
    resultado = pd.read_sql_query(consulta, conn)
    print("Esquema da tabela:", tabela)
    display(resultado)
    print("-"*100)
    print("\n")
```

```
Esquema da tabela: people
```

| | cid | name | type | notnull | dflt_value | pk |
|---|---|---|---|---|---|---|
| 0 | 0 | person_id | VARCHAR | 0 | None | 1 |
| 1 | 1 | name | VARCHAR | 0 | None | 0 |
| 2 | 2 | born | INTEGER | 0 | None | 0 |
| 3 | 3 | died | INTEGER | 0 | None | 0 |

```
--------------------------------------------------------------------------------
------------------------------


Esquema da tabela: titles
```

| | cid | name | type | notnull | dflt_value | pk |
|---|---|---|---|---|---|---|
| 0 | 0 | title_id | VARCHAR | 0 | None | 1 |
| 1 | 1 | type | VARCHAR | 0 | None | 0 |
| 2 | 2 | primary_title | VARCHAR | 0 | None | 0 |
| 3 | 3 | original_title | VARCHAR | 0 | None | 0 |
| 4 | 4 | is_adult | INTEGER | 0 | None | 0 |
| 5 | 5 | premiered | INTEGER | 0 | None | 0 |
| 6 | 6 | ended | INTEGER | 0 | None | 0 |
| 7 | 7 | runtime_minutes | INTEGER | 0 | None | 0 |
| 8 | 8 | genres | VARCHAR | 0 | None | 0 |

```
--------------------------------------------------------------------------------
------------------------------


Esquema da tabela: akas
```

| | cid | name | type | notnull | dflt_value | pk |
|---|---|---|---|---|---|---|
| 0 | 0 | title_id | VARCHAR | 0 | None | 0 |
| 1 | 1 | title | VARCHAR | 0 | None | 0 |
| 2 | 2 | region | VARCHAR | 0 | None | 0 |
| 3 | 3 | language | VARCHAR | 0 | None | 0 |
| 4 | 4 | types | VARCHAR | 0 | None | 0 |

```
5    5         attributes  VARCHAR        0       None   0
6    6  is_original_title  INTEGER        0       None   0

------------------------------------------------------------------------
------------------------------


Esquema da tabela: crew

   cid        name     type  notnull dflt_value  pk
0    0    title_id  VARCHAR        0       None   0
1    1   person_id  VARCHAR        0       None   0
2    2    category  VARCHAR        0       None   0
3    3         job  VARCHAR        0       None   0
4    4  characters  VARCHAR        0       None   0

------------------------------------------------------------------------
------------------------------


Esquema da tabela: episodes

   cid              name     type  notnull dflt_value  pk
0    0  episode_title_id  INTEGER        0       None   0
1    1     show_title_id  INTEGER        0       None   0
2    2     season_number  INTEGER        0       None   0
3    3    eposide_number  INTEGER        0       None   0

------------------------------------------------------------------------
------------------------------


Esquema da tabela: ratings

   cid      name     type  notnull dflt_value  pk
0    0  title_id  VARCHAR        0       None   1
1    1    rating  INTEGER        0       None   0
2    2     votes  INTEGER        0       None   0

------------------------------------------------------------------------
------------------------------



#1.                        Quais são as categorias de filmes mais
comuns no imdb

# Cria a consulta SQL
consulta1 = '''SELECT type, COUNT(*) AS COUNT FROM titles GROUP BY
type'''
```

```
# Extrai o resultado
resultado1 = pd.read_sql_query(consulta1, conn)

# Visualiza o resultado
display(resultado1)
```

```
            type    COUNT
0          movie   621980
1          short   890751
2      tvEpisode  6970082
3   tvMiniSeries    45241
4        tvMovie   137621
5        tvPilot        2
6       tvSeries   231477
7        tvShort    10673
8      tvSpecial    38454
9          video   265452
10     videoGame    32292
```

```
# Vamos calcular o percentual para cada tipo
resultado1['percentual'] = (resultado1['COUNT'] /
resultado1['COUNT'].sum()) * 100

# Visualiza o resultado
display(resultado1)
```

```
            type    COUNT  percentual
0          movie   621980    6.728454
1          short   890751    9.635965
2      tvEpisode  6970082   75.400943
3   tvMiniSeries    45241    0.489408
4        tvMovie   137621    1.488756
5        tvPilot        2    0.000022
6       tvSeries   231477    2.504072
7        tvShort    10673    0.115458
8      tvSpecial    38454    0.415988
9          video   265452    2.871606
10     videoGame    32292    0.349328
```

```
# Vamos criar um gráfico com apenas 4 categorias:
# As 3 categorias com mais títulos e 1 categoria com todo o restante

# Cria um dicionário vazio
others = {}

# Filtra o percentual em 5% e soma o total
others['COUNT'] = resultado1[resultado1['percentual'] < 5]
['COUNT'].sum()

# Grava o percentual
others['percentual'] = resultado1[resultado1['percentual'] < 5]
```

```
['percentual'].sum()

# Ajusta o nome
others['type'] = 'others'


# Visualiza
others
```

```
{'COUNT': 761212, 'percentual': 8.234638049983637, 'type': 'others'}
```

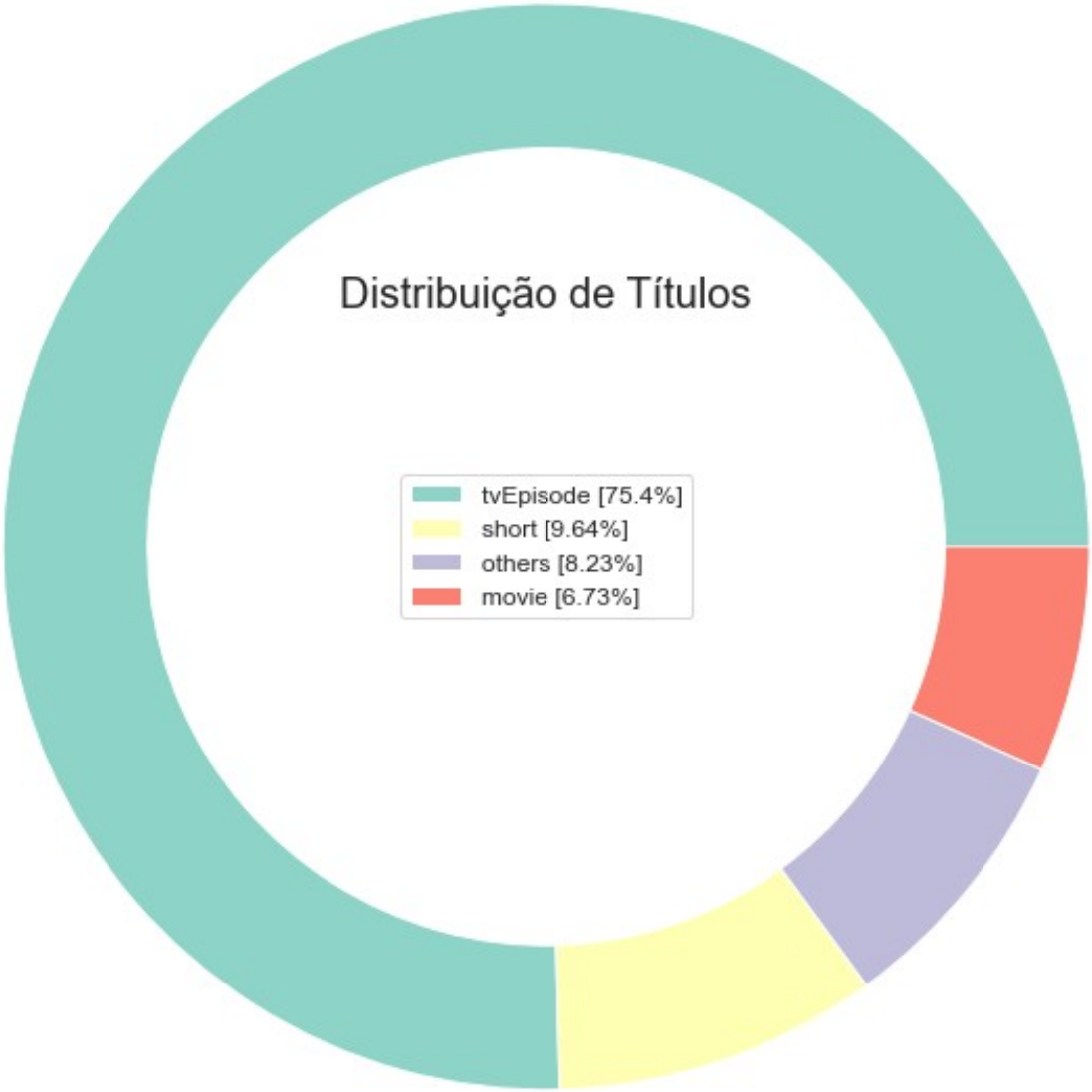
```
# Filtra o dataframe de resultado
resultado1 = resultado1[resultado1['percentual'] > 6]

# Append com o dataframe de outras categorias
resultado1 = resultado1.append(others, ignore_index = True)

# Ordena o resultado
resultado1 = resultado1.sort_values(by = 'COUNT', ascending = False)

# Visualiza
resultado1.head(10)
```

```
        type    COUNT  percentual
2  tvEpisode  6970082   75.400943
1      short   890751    9.635965
3     others   761212    8.234638
0      movie   621980    6.728454
```

```
# Ajusta os labels           List Compreension
labels = [str(resultado1['type'][i])+'
'+'['+str(round(resultado1['percentual'][i],2)) +'%'+']' for i in
resultado1.index]

# Plot

# Mapa de cores
# https://matplotlib.org/stable/tutorials/colors/colormaps.html
cs = cm.Set3(np.arange(100))

# Cria a figura
f = plt.figure()
```

```
<Figure size 432x288 with 0 Axes>
```

```
# Pie Plot
plt.pie(resultado1['COUNT'], labeldistance = 1, radius = 3, colors =
cs, wedgeprops = dict(width = 0.8))
plt.legend(labels = labels, loc = 'center', prop = {'size':12})
plt.title("Distribuição de Títulos", loc = 'Center', fontdict =
{'fontsize':20,'fontweight':20})
plt.show()
```




```
#2.                                   Qual o número de títulos por genero

# Cria a consulta SQL
consulta2 = '''SELECT genres, COUNT(*) FROM titles WHERE type =
'movie' GROUP BY genres'''

# Resultado
resultado2 = pd.read_sql_query(consulta2, conn)

# Visualiza o resultado
display(resultado2)
```

| | genres | COUNT(*) |
|---|---|---|
| 0 | Action | 14164 |
| 1 | Action,Adult | 11 |
| 2 | Action,Adult,Adventure | 2 |
| 3 | Action,Adult,Comedy | 6 |

```
4         Action,Adult,Crime         9
...                      ...       ...
1463        Thriller,Western        42
1464                     War      1312
1465             War,Western        14
1466                 Western      5153
1467                      \N     71825

[1468 rows x 2 columns]
```

```
# Converte as strings para minúsculo
resultado2['genres'] = resultado2['genres'].str.lower().values

# Remove valores NA (ausentes)
temp = resultado2['genres'].dropna()

# Vamos criar um vetor usando expressão regular para filtrar as
strings

# https://docs.python.org/3.8/library/re.html
padrao = '(?u)\\b[\\w-]+\\b'

#https://scikit-learn.org/stable/modules/generated/
sklearn.feature_extraction.text.CountVectorizer.html
vetor = CountVectorizer(token_pattern = padrao, analyzer =
'word').fit(temp)
type(vetor)
```

```
sklearn.feature_extraction.text.CountVectorizer
```

```
# Aplica a vetorização ao dataset sem valores NA
bag_generos = vetor.transform(temp)

type(bag_generos)
```

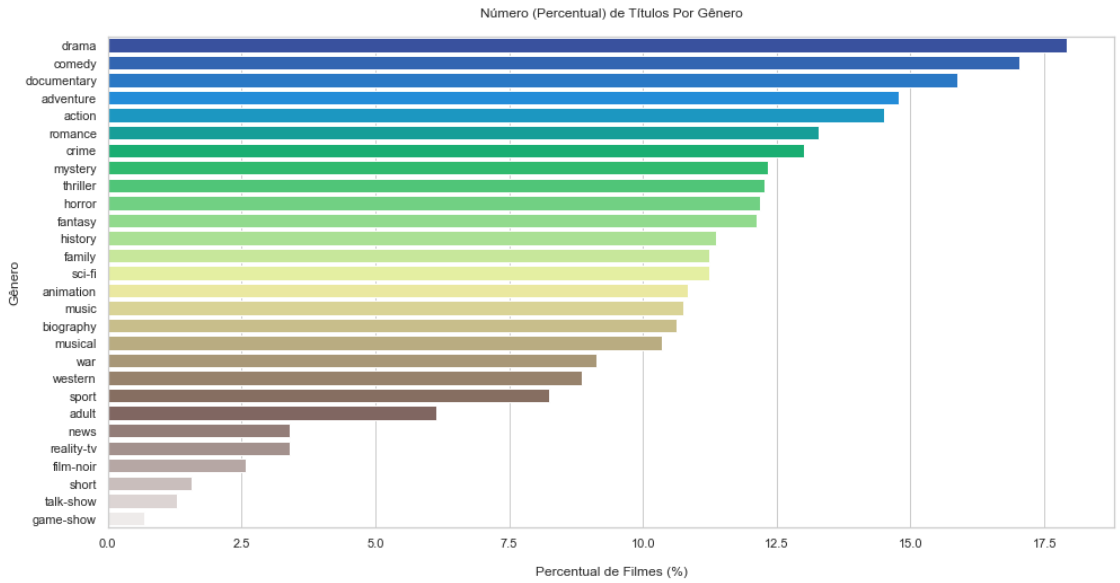
```
scipy.sparse.csr.csr_matrix
```

```
# Retorna gêneros únicos
generos_unicos =  vetor.get_feature_names()

# Cria o dataframe de gêneros
generos = pd.DataFrame(bag_generos.todense(), columns =
generos_unicos, index = temp.index)

# Drop da coluna n
generos = generos.drop(columns = 'n', axis = 0)

# Calcula o percentual
generos_percentual = 100 *
pd.Series(generos.sum()).sort_values(ascending = False) /
generos.shape[0]
```

```
# Visualiza
generos_percentual.head(10)
```

```
drama          17.915531
comedy         17.029973
documentary    15.871935
adventure      14.782016
action         14.509537
romance        13.283379
crime          13.010899
mystery        12.329700
thriller       12.261580
horror         12.193460
dtype: float64
```

```
# Plot
plt.figure(figsize = (16,8))
sns.barplot(x = generos_percentual.values, y =
generos_percentual.index, orient = "h", palette = "terrain")
plt.ylabel('Gênero')
plt.xlabel("\nPercentual de Filmes (%)")
plt.title('\nNúmero (Percentual) de Títulos Por Gênero\n')
plt.show()
```




```
#3.                                  Qual é a mediana de avaliação dos
Filmes por genero?
#pq nao a media? pq a mediana não é afetada por outlyers

# Consulta SQL
consulta3 = '''
            SELECT rating, genres FROM
            ratings JOIN titles ON ratings.title_id = titles.title_id
            WHERE premiered <= 2022 AND type = 'movie' '''
```

```
# Resultado
resultado3 = pd.read_sql_query(consulta3, conn)
# Visualiza
display(resultado3)
```

```
        rating                       genres
0          5.2                      Romance
1          3.7                           \N
2          6.0  Action,Adventure,Biography
3          4.0                        Drama
4          4.1                        Drama
...        ...                          ...
282568     5.8                     Thriller
282569     6.4                Drama,History
282570     3.8       Adventure,History,War
282571     8.3                        Drama
282572     8.4                           \N

[282573 rows x 2 columns]
```

```
# Vamos criar uma função para retornar os genêros
def retorna_generos(df):
    df['genres'] = df['genres'].str.lower().values
    temp = df['genres'].dropna()
    vetor = CountVectorizer(token_pattern = '(?u)\\b[\\w-]+\\b',
analyzer = 'word').fit(temp)
    generos_unicos =  vetor.get_feature_names()
    generos_unicos = [genre for genre in generos_unicos if len(genre)
> 1]
    return generos_unicos

# Aplica a função
generos_unicos = retorna_generos(resultado3)
# Visualiza
generos_unicos
```

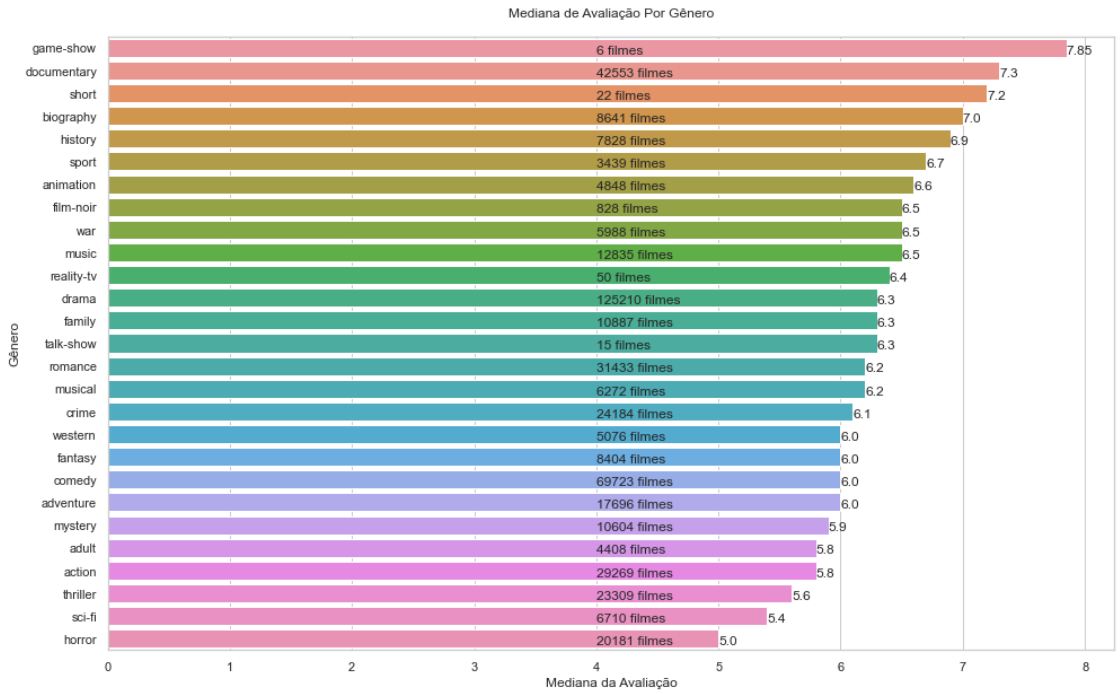
```
['action',
 'adult',
 'adventure',
 'animation',
 'biography',
 'comedy',
 'crime',
 'documentary',
 'drama',
 'family',
 'fantasy',
 'film-noir',
 'game-show',
 'history',
 'horror',
```

```
 'music',
 'musical',
 'mystery',
 'news',
 'reality-tv',
 'romance',
 'sci-fi',
 'short',
 'sport',
 'talk-show',
 'thriller',
 'war',
 'western']

# Cria listas vazias
genero_counts = []
genero_ratings = []

# Loop
for item in generos_unicos:
    # Retorna a contagem de filmes por gênero
    consulta = 'SELECT COUNT(rating) FROM ratings JOIN titles ON
ratings.title_id=titles.title_id WHERE genres LIKE '+
'\''+'%'+item+'%'+'\' AND type=\'movie\''
    resultado = pd.read_sql_query(consulta, conn)
    genero_counts.append(resultado.values[0][0])

     # Retorna a avaliação de filmes por gênero
    consulta = 'SELECT rating FROM ratings JOIN titles ON
ratings.title_id=titles.title_id WHERE genres LIKE '+
'\''+'%'+item+'%'+'\' AND type=\'movie\''
    resultado = pd.read_sql_query(consulta, conn)
    genero_ratings.append(np.median(resultado['rating']))

# Prepara o dataframe final
df_genero_ratings = pd.DataFrame()
df_genero_ratings['genres'] = generos_unicos
df_genero_ratings['count'] = genero_counts
df_genero_ratings['rating'] = genero_ratings

# Visualiza
df_genero_ratings.head(20)
```

| | genres | count | rating |
|---|---|---|---|
| 0 | action | 29269 | 5.80 |
| 1 | adult | 4408 | 5.80 |
| 2 | adventure | 17696 | 6.00 |
| 3 | animation | 4848 | 6.60 |
| 4 | biography | 8641 | 7.00 |
| 5 | comedy | 69723 | 6.00 |
| 6 | crime | 24184 | 6.10 |

```
7   documentary   42553    7.30
8         drama  125210    6.30
9        family   10887    6.30
10      fantasy    8404    6.00
11    film-noir     828    6.50
12    game-show       6    7.85
13      history    7828    6.90
14       horror   20181    5.00
15        music   12835    6.50
16      musical    6272    6.20
17      mystery   10604    5.90
18         news     658    7.30
19   reality-tv      50    6.40
```

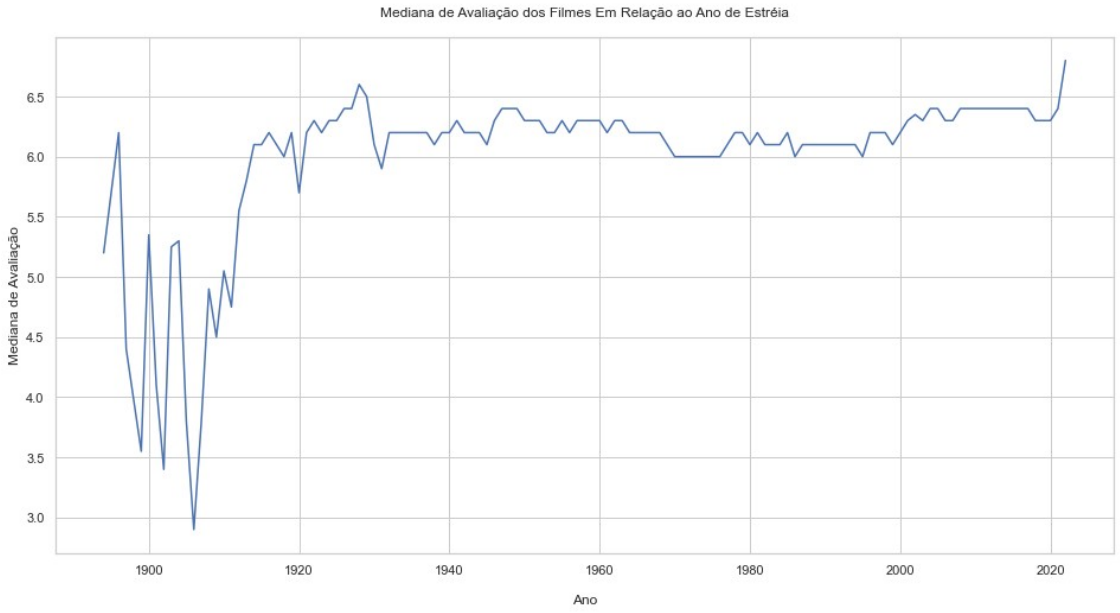
```
# Drop do índice 18 (news)
# Não queremos essa informação como gênero
df_genero_ratings = df_genero_ratings.drop(index = 18)

# Ordena o resultado
df_genero_ratings = df_genero_ratings.sort_values(by = 'rating',
ascending = False)

# Plot

# Figura
plt.figure(figsize = (16,10))

# Barplot
sns.barplot(y = df_genero_ratings.genres, x =
df_genero_ratings.rating, orient = "h")

# Textos do gráfico
for i in range(len(df_genero_ratings.index)):

    plt.text(4.0,
             i + 0.25,
             str(df_genero_ratings['count']
[df_genero_ratings.index[i]]) + " filmes")

    plt.text(df_genero_ratings.rating[df_genero_ratings.index[i]],
             i + 0.25,
             round(df_genero_ratings["rating"]
[df_genero_ratings.index[i]],2))

plt.ylabel('Gênero')
plt.xlabel('Mediana da Avaliação')
plt.title('\nMediana de Avaliação Por Gênero\n')
plt.show()
```




```
#4.                                  Qual é a mediana de avaliação dos
filmes ao ano de estréia?

# Consulta SQL
consulta4 = '''
            SELECT rating AS Rating, premiered FROM
            ratings JOIN titles ON ratings.title_id = titles.title_id
            WHERE premiered <= 2022 AND type = 'movie'
            ORDER BY premiered
            '''

# Resultado
resultado4 = pd.read_sql_query(consulta4, conn)
display(resultado4)
```

```
        Rating  premiered
0          5.2       1894
1          6.2       1896
2          4.4       1897
3          3.9       1899
4          3.2       1899
...        ...        ...
282568     8.2       2022
282569     6.8       2022
282570     5.3       2022
282571     3.9       2022
282572     4.0       2022

[282573 rows x 2 columns]
```

```
# Calculamos a mediana ao longo do tempo (anos)
ratings = []
for year in set(resultado4['premiered']):
    ratings.append(np.median(resultado4[resultado4['premiered'] ==
year]['Rating']))

type(ratings)

list

ratings[1:10]

[6.2, 4.4, 3.55, 5.35, 4.1, 3.4, 5.25, 5.3, 3.8]

# Lista de anos
anos = list(set(resultado4['premiered']))

anos[1:10]

[1896, 1897, 1899, 1900, 1901, 1902, 1903, 1904, 1905]

# Plot
plt.figure(figsize = (16,8))
plt.plot(anos, ratings)
plt.xlabel('\nAno')
plt.ylabel('Mediana de Avaliação')
plt.title('\nMediana de Avaliação dos Filmes Em Relação ao Ano de
Estréia\n')
plt.show()
```



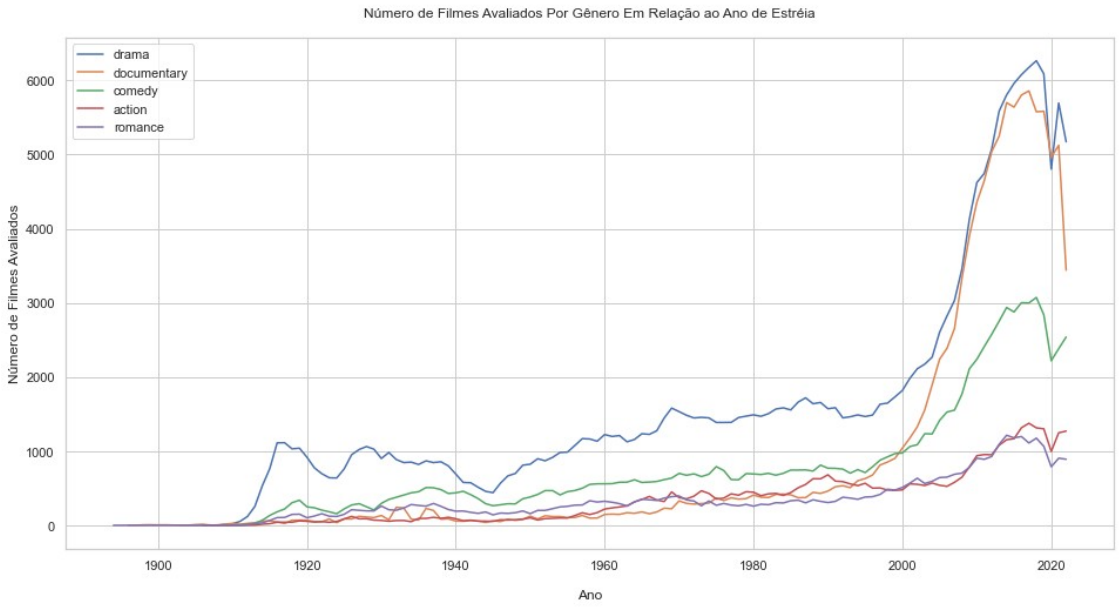

*#5. Qual o numero de filmes avaliado por ano de estreia?*

```
# Consulta SQL
consulta5 = '''SELECT genres FROM titles '''

# Resultado
resultado5 = pd.read_sql_query(consulta5, conn)
display(resultado5)
```

```
                          genres
0                Documentary,Short
1                  Animation,Short
2          Animation,Comedy,Romance
3                  Animation,Short
4                     Comedy,Short
...                            ...
9244020         Action,Drama,Family
9244021         Action,Drama,Family
9244022         Action,Drama,Family
9244023                       Short
9244024  Adventure,Animation,Comedy

[9244025 rows x 1 columns]
```

```
# Retorna gêneros únicos
generos_unicos = retorna_generos(resultado5)
# Visualiza o resultado
generos_unicos

# Agora fazemos a contagem
genero_count = []
for item in generos_unicos:
    consulta = 'SELECT COUNT(*) COUNT FROM  titles  WHERE genres LIKE
'+ '\''+'%'+item+'%'+'\' AND type=\'movie\' AND premiered <= 2022'
    resultado = pd.read_sql_query(consulta, conn)
    genero_count.append(resultado['COUNT'].values[0])

# Prepara o dataframe
df_genero_count = pd.DataFrame()
df_genero_count['genre'] = generos_unicos
df_genero_count['Count'] = genero_count

# Calcula os top 5
df_genero_count = df_genero_count[df_genero_count['genre'] != 'n']
df_genero_count = df_genero_count.sort_values(by = 'Count', ascending
= False)
top_generos = df_genero_count.head()['genre'].values

# Plot

# Figura
plt.figure(figsize = (16,8))
```

```
# Loop e Plot
for item in top_generos:
    consulta = 'SELECT COUNT(*) Number_of_movies, premiered Year FROM
titles  WHERE genres LIKE '+ '\''+'%'+item+'%'+'\' AND type=\'movie\'
AND Year <=2022 GROUP BY Year'
    resultado = pd.read_sql_query(consulta, conn)
    plt.plot(resultado['Year'], resultado['Number_of_movies'])

plt.xlabel('\nAno')
plt.ylabel('Número de Filmes Avaliados')
plt.title('\nNúmero de Filmes Avaliados Por Gênero Em Relação ao Ano
de Estréia\n')
plt.legend(labels = top_generos)
plt.show()
```




```
#6.                  Qual o filme com maior tempo de duração? Calculo
do percentis

# Consulta SQL
consulta6 = '''
            SELECT runtime_minutes Runtime
            FROM titles
            WHERE type = 'movie' AND Runtime != 'NaN'
            '''

# Resultado
resultado6 = pd.read_sql_query(consulta6, conn)

display(resultado6)
```

| | Runtime |
|---|---|
| 0 | 45 |

```
1           100
2            70
3            90
4           120
...         ...
392525      123
392526       57
392527      100
392528      116
392529       49

[392530 rows x 1 columns]
```

```
# Loop para cálculo dos percentis
for i in range(101):
    val = i
    perc = round(np.percentile(resultado6['Runtime'].values, val), 2)
    print('{} percentil da duração (runtime) é: {}'.format(val, perc))
```

```
0 percentil da duração (runtime) é: 1.0
1 percentil da duração (runtime) é: 45.0
2 percentil da duração (runtime) é: 47.0
3 percentil da duração (runtime) é: 50.0
4 percentil da duração (runtime) é: 50.0
5 percentil da duração (runtime) é: 51.0
6 percentil da duração (runtime) é: 52.0
7 percentil da duração (runtime) é: 53.0
8 percentil da duração (runtime) é: 54.0
9 percentil da duração (runtime) é: 56.0
10 percentil da duração (runtime) é: 57.0
11 percentil da duração (runtime) é: 58.0
12 percentil da duração (runtime) é: 60.0
13 percentil da duração (runtime) é: 60.0
14 percentil da duração (runtime) é: 60.0
15 percentil da duração (runtime) é: 61.0
16 percentil da duração (runtime) é: 62.0
17 percentil da duração (runtime) é: 64.0
18 percentil da duração (runtime) é: 65.0
19 percentil da duração (runtime) é: 67.0
20 percentil da duração (runtime) é: 69.0
21 percentil da duração (runtime) é: 70.0
22 percentil da duração (runtime) é: 70.0
23 percentil da duração (runtime) é: 71.0
24 percentil da duração (runtime) é: 72.0
25 percentil da duração (runtime) é: 73.0
26 percentil da duração (runtime) é: 74.0
27 percentil da duração (runtime) é: 75.0
28 percentil da duração (runtime) é: 76.0
29 percentil da duração (runtime) é: 77.0
30 percentil da duração (runtime) é: 78.0
```

31 percentil da duração (runtime) é: 78.0
32 percentil da duração (runtime) é: 79.0
33 percentil da duração (runtime) é: 80.0
34 percentil da duração (runtime) é: 80.0
35 percentil da duração (runtime) é: 80.0
36 percentil da duração (runtime) é: 81.0
37 percentil da duração (runtime) é: 82.0
38 percentil da duração (runtime) é: 83.0
39 percentil da duração (runtime) é: 83.0
40 percentil da duração (runtime) é: 84.0
41 percentil da duração (runtime) é: 85.0
42 percentil da duração (runtime) é: 85.0
43 percentil da duração (runtime) é: 85.0
44 percentil da duração (runtime) é: 86.0
45 percentil da duração (runtime) é: 86.0
46 percentil da duração (runtime) é: 87.0
47 percentil da duração (runtime) é: 87.0
48 percentil da duração (runtime) é: 88.0
49 percentil da duração (runtime) é: 88.0
50 percentil da duração (runtime) é: 89.0
51 percentil da duração (runtime) é: 89.0
52 percentil da duração (runtime) é: 90.0
53 percentil da duração (runtime) é: 90.0
54 percentil da duração (runtime) é: 90.0
55 percentil da duração (runtime) é: 90.0
56 percentil da duração (runtime) é: 90.0
57 percentil da duração (runtime) é: 90.0
58 percentil da duração (runtime) é: 91.0
59 percentil da duração (runtime) é: 91.0
60 percentil da duração (runtime) é: 92.0
61 percentil da duração (runtime) é: 92.0
62 percentil da duração (runtime) é: 93.0
63 percentil da duração (runtime) é: 93.0
64 percentil da duração (runtime) é: 94.0
65 percentil da duração (runtime) é: 94.0
66 percentil da duração (runtime) é: 95.0
67 percentil da duração (runtime) é: 95.0
68 percentil da duração (runtime) é: 96.0
69 percentil da duração (runtime) é: 96.0
70 percentil da duração (runtime) é: 97.0
71 percentil da duração (runtime) é: 98.0
72 percentil da duração (runtime) é: 98.0
73 percentil da duração (runtime) é: 99.0
74 percentil da duração (runtime) é: 100.0
75 percentil da duração (runtime) é: 100.0
76 percentil da duração (runtime) é: 100.0
77 percentil da duração (runtime) é: 101.0
78 percentil da duração (runtime) é: 102.0
79 percentil da duração (runtime) é: 103.0
80 percentil da duração (runtime) é: 104.0

```
81 percentil da duração (runtime) é: 105.0
82 percentil da duração (runtime) é: 106.0
83 percentil da duração (runtime) é: 107.0
84 percentil da duração (runtime) é: 108.0
85 percentil da duração (runtime) é: 110.0
86 percentil da duração (runtime) é: 111.0
87 percentil da duração (runtime) é: 113.0
88 percentil da duração (runtime) é: 115.0
89 percentil da duração (runtime) é: 117.0
90 percentil da duração (runtime) é: 119.0
91 percentil da duração (runtime) é: 120.0
92 percentil da duração (runtime) é: 123.0
93 percentil da duração (runtime) é: 126.0
94 percentil da duração (runtime) é: 130.0
95 percentil da duração (runtime) é: 135.0
96 percentil da duração (runtime) é: 139.0
97 percentil da duração (runtime) é: 145.0
98 percentil da duração (runtime) é: 153.0
99 percentil da duração (runtime) é: 168.0
100 percentil da duração (runtime) é: 51420.0
```

```
# Refazendo a consulta e retornando o filme com maior duração
consulta6 = '''
            SELECT runtime_minutes Runtime, primary_title
            FROM titles
            WHERE type = 'movie' AND Runtime != 'NaN'
            ORDER BY Runtime DESC
            LIMIT 1
            '''

resultado6 = pd.read_sql_query(consulta6, conn)
resultado6
```

```
   Runtime primary_title
0    51420     Logistics
```

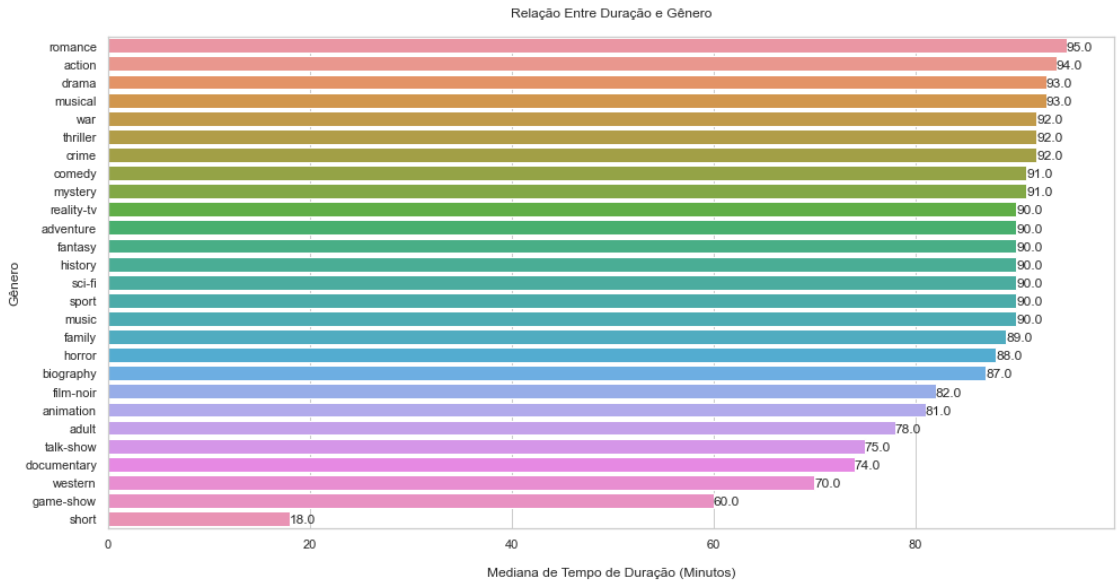
```
#7.                                  Qual é a relação entr duração e
genero?

# Consulta SQL
consulta7 = '''
            SELECT AVG(runtime_minutes) Runtime, genres
            FROM titles
            WHERE type = 'movie'
            AND runtime_minutes != 'NaN'
            GROUP BY genres
            '''

# Resultado
resultado7 = pd.read_sql_query(consulta7, conn)
resultado7
```

```
        Runtime                  genres
0     99.315331                  Action
1     77.000000            Action,Adult
2     85.000000  Action,Adult,Adventure
3     76.400000     Action,Adult,Comedy
4     85.375000      Action,Adult,Crime
...         ...                     ...
1380  95.266667        Thriller,Western
1381  93.847650                     War
1382  90.000000             War,Western
1383  69.992075                 Western
1384  82.472098                      \N

[1385 rows x 2 columns]
```

```
# Retorna gêneros únicos
generos_unicos = retorna_generos(resultado7)

# Visualiza
generos_unicos
```

```
['action',
 'adult',
 'adventure',
 'animation',
 'biography',
 'comedy',
 'crime',
 'documentary',
 'drama',
 'family',
 'fantasy',
 'film-noir',
 'game-show',
 'history',
 'horror',
 'music',
 'musical',
 'mystery',
 'news',
 'reality-tv',
 'romance',
 'sci-fi',
 'short',
 'sport',
 'talk-show',
 'thriller',
 'war',
 'western']
```

```
# Calcula duração por gênero
genero_runtime = []
for item in generos_unicos:
    consulta = 'SELECT runtime_minutes Runtime FROM  titles  WHERE
genres LIKE '+ '\''+'%'+item+'%'+'\' AND type=\'movie\' AND Runtime!
=\'NaN\''
    resultado = pd.read_sql_query(consulta, conn)
    genero_runtime.append(np.median(resultado['Runtime']))

# Prepara o dataframe
df_genero_runtime = pd.DataFrame()
df_genero_runtime['genre'] = generos_unicos
df_genero_runtime['runtime'] = genero_runtime

# Remove índice 18 (news)
df_genero_runtime = df_genero_runtime.drop(index = 18)

# Ordena os dados
df_genero_runtime = df_genero_runtime.sort_values(by = 'runtime',
ascending = False)

# Plot

# Tamanho da figura
plt.figure(figsize = (16,8))

# Barplot
sns.barplot(y = df_genero_runtime.genre, x =
df_genero_runtime.runtime, orient = "h")
# Loop
for i in range(len(df_genero_runtime.index)):
    plt.text(df_genero_runtime.runtime[df_genero_runtime.index[i]],
             i + 0.25,
             round(df_genero_runtime["runtime"]
[df_genero_runtime.index[i]], 2))

plt.ylabel('Gênero')
plt.xlabel('\nMediana de Tempo de Duração (Minutos)')
plt.title('\nRelação Entre Duração e Gênero\n')
plt.show()
```




```
#8.                          Qual o número de filmes produzidos por
país?

# Consulta SQL
consulta8 = '''
            SELECT region, COUNT(*) Number_of_movies FROM
            akas JOIN titles ON
            akas.title_id = titles.title_id
            WHERE region != 'None'
            AND type = \'movie\'
            GROUP BY region
            '''

# Resultado
resultado8 = pd.read_sql_query(consulta8, conn)

display(resultado8)
```

```
    region  Number_of_movies
0       AD                21
1       AE              8158
2       AF               111
3       AG                12
4       AL              1303
..     ...               ...
230   YUCS               152
231     ZA              8116
232     ZM                47
233   ZRCD                 2
234     ZW                56

[235 rows x 2 columns]
```

```
# Shape
resultado8.shape
```

```
(235, 2)
```

```
# Número de linhas
resultado8.shape[0]
```

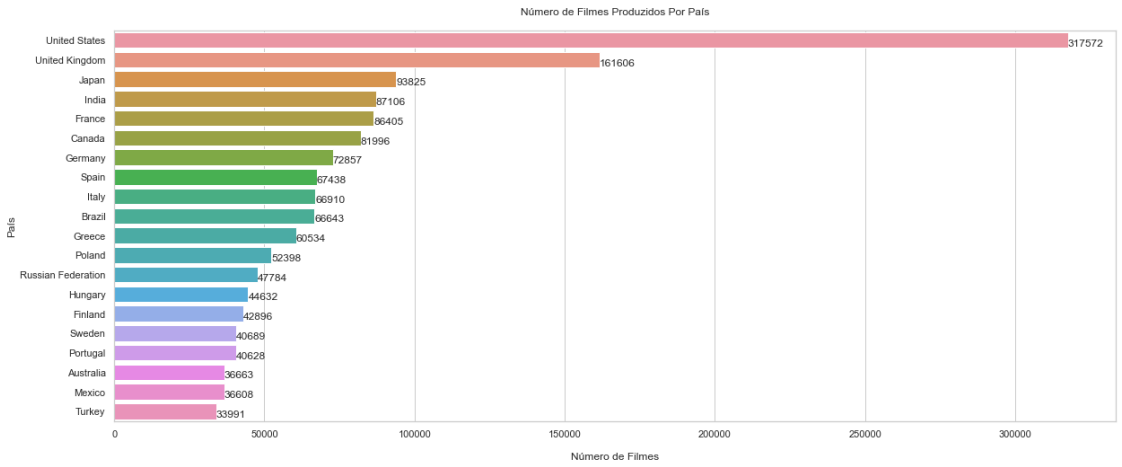
```
235
```

```
# Listas auxiliares
nomes_paises = []
contagem = []

# Loop para obter o país de acordo com a região
for i in range(resultado8.shape[0]):
    try:
        coun = resultado8['region'].values[i]
        nomes_paises.append(pycountry.countries.get(alpha_2 =
coun).name)
        contagem.append(resultado8['Number_of_movies'].values[i])
    except:
        continue

# Prepara o dataframe
df_filmes_paises = pd.DataFrame()
df_filmes_paises['country'] = nomes_paises
df_filmes_paises['Movie_Count'] = contagem

# Ordena o resultado
df_filmes_paises = df_filmes_paises.sort_values(by = 'Movie_Count',
ascending = False)

# Visualiza
df_filmes_paises.head(10)
```

| | country | Movie_Count |
|---|---|---|
| 200 | United States | 317572 |
| 66 | United Kingdom | 161606 |
| 97 | Japan | 93825 |
| 90 | India | 87106 |
| 64 | France | 86405 |
| 32 | Canada | 81996 |
| 48 | Germany | 72857 |
| 59 | Spain | 67438 |
| 94 | Italy | 66910 |
| 26 | Brazil | 66643 |

```
# Plot

# Figura
plt.figure(figsize = (20,8))
```

```
# Barplot
sns.barplot(y = df_filmes_paises[:20].country, x =
df_filmes_paises[:20].Movie_Count, orient = "h")

# Loop
for i in range(0,20):
    plt.text(df_filmes_paises.Movie_Count[df_filmes_paises.index[i]]-
1,
             i + 0.30,
             round(df_filmes_paises["Movie_Count"]
[df_filmes_paises.index[i]],2))

plt.ylabel('País')
plt.xlabel('\nNúmero de Filmes')
plt.title('\nNúmero de Filmes Produzidos Por País\n')
plt.show()
```




```
#9.                          Quais são os top 10 melhores filmes?

# Consulta SQL
consulta9 = '''
            SELECT primary_title AS Movie_Name, genres, rating
            FROM
            titles JOIN ratings
            ON  titles.title_id = ratings.title_id
            WHERE titles.type = 'movie' AND ratings.votes >= 25000
            ORDER BY rating DESC
            LIMIT 10
            '''

# Resultado
top10_melhores_filmes = pd.read_sql_query(consulta9, conn)

display(top10_melhores_filmes)
```

```
                                    Movie_Name
genres  \
0                      The Shawshank Redemption
Drama
1                                The Godfather
Crime,Drama
2                             The Chaos Class
Comedy,Drama
3                       CM101MMXI Fundamentals
Comedy,Documentary
4                                 12 Angry Men
Crime,Drama
5                        The Godfather Part II
Crime,Drama
6                             Schindler's List
Biography,Drama,History
7  The Lord of the Rings: The Return of the King
Action,Adventure,Drama
8                             The Dark Knight
Action,Crime,Drama
9                                 Mirror Game
Crime,Mystery,Thriller

   rating
0     9.3
1     9.2
2     9.2
3     9.1
4     9.0
5     9.0
6     9.0
7     9.0
8     9.0
9     9.0
```

```
#10.                  Quais são os top 10 piores filmes?

# Consulta SQL
consulta10 = '''
             SELECT primary_title AS Movie_Name, genres, rating
             FROM
             titles JOIN ratings
             ON  titles.title_id = ratings.title_id
             WHERE titles.type = 'movie' AND ratings.votes >= 25000
             ORDER BY rating ASC
             LIMIT 10
             '''

# Resultado
top10_piores_filmes = pd.read_sql_query(consulta10, conn)
```

```
display(top10_piores_filmes)
```

| | Movie_Name | genres | rating |
|---|---|---|---|
| 0 | Reis | Biography,Drama | 1.0 |
| 1 | Cumali Ceber | Comedy | 1.0 |
| 2 | Sadak 2 | Action,Drama | 1.1 |
| 3 | Smolensk | Drama,Thriller | 1.2 |
| 4 | Superbabies: Baby Geniuses 2 | Comedy,Family,Sci-Fi | 1.5 |
| 5 | The Cost of Deception | Crime,Drama,History | 1.5 |
| 6 | Manos: The Hands of Fate | Horror | 1.6 |
| 7 | Justin Bieber: Never Say Never | Documentary,Music | 1.6 |
| 8 | From Justin to Kelly | Comedy,Musical,Romance | 1.9 |
| 9 | The Hottie & the Nottie | Comedy,Romance | 1.9 |