In [ ]:
```
import pandas as pd
```

In [ ]:
```
df = pd.read_csv('desafio_indicium_imdb.csv')
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 16 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Unnamed: 0     999 non-null    int64  
 1   Series_Title   999 non-null    object 
 2   Released_Year  999 non-null    object 
 3   Certificate    898 non-null    object 
 4   Runtime        999 non-null    object 
 5   Genre          999 non-null    object 
 6   IMDB_Rating    999 non-null    float64
 7   Overview       999 non-null    object 
 8   Meta_score     842 non-null    float64
 9   Director       999 non-null    object 
 10  Star1          999 non-null    object 
 11  Star2          999 non-null    object 
 12  Star3          999 non-null    object 
 13  Star4          999 non-null    object 
 14  No_of_Votes    999 non-null    int64  
 15  Gross          830 non-null    object 
dtypes: float64(2), int64(2), object(12)
memory usage: 125.0+ KB
```

In [ ]:
```
df[df['Gross'].isna()]
```

Out[ ]:

	Unnamed: 0	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rati
17	18	Hamilton	2020	PG-13	160 min	Biography, Drama, History	
19	20	Soorarai Pottru	2020	U	153 min	Drama	
29	30	Seppuku	1962	NaN	133 min	Action, Drama, Mystery	
31	32	It's a Wonderful Life	1946	PG	130 min	Drama, Family, Fantasy	
45	46	Hotaru no haka	1988	U	89 min	Animation, Drama, War	
...	...	...	...	...	...	...	
992	993	Blowup	1966	A	111 min	Drama, Mystery, Thriller	
994	995	Breakfast at Tiffany's	1961	A	115 min	Comedy, Drama, Romance	
995	996	Giant	1956	G	201 min	Drama, Western	
997	998	Lifeboat	1944	NaN	97 min	Drama, War	
998	999	The 39 Steps	1935	NaN	86 min	Crime, Mystery, Thriller	

169 rows × 16 columns

In [ ]:
```
df[df['Genre'].isin(['Drama'])]
```

Out[ ]:

	Unnamed: 0	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating
8	9	Fight Club	1999	A	139 min	Drama	8.8
16	17	One Flew Over the Cuckoo's Nest	1975	A	133 min	Drama	8.7
19	20	Soorarai Pottru	2020	U	153 min	Drama	8.6
39	40	American History X	1998	R	119 min	Drama	8.5
52	53	Capharnaüm	2018	A	126 min	Drama	8.4
...	...	...	...	...	...	...	...
932	933	Synecdoche, New York	2008	R	124 min	Drama	7.6
933	934	Mysterious Skin	2004	R	105 min	Drama	7.6
940	941	25th Hour	2002	R	135 min	Drama	7.6
945	946	Y tu mamá también	2001	A	106 min	Drama	7.6
980	981	On Golden Pond	1981	UA	109 min	Drama	7.6

	Unnamed: 0	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating

84 rows × 16 columns

In [ ]:
```
df[df['Certificate'].isna()]
```

Out[ ]:

	Unnamed: 0	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rati
29	30	Seppuku	1962	NaN	133 min	Action, Drama, Mystery	
53	54	Ayla: The Daughter of War	2017	NaN	125 min	Biography, Drama, History	
76	77	Tengoku to jigoku	1963	NaN	143 min	Crime, Drama, Mystery	
91	92	Babam ve Oglum	2005	NaN	112 min	Drama, Family	
120	121	Ikiru	1952	NaN	143 min	Drama	
...	...	...	...	...	...	...	
919	920	The Secret of Kells	2009	NaN	71 min	Animation, Adventure, Family	
925	926	Dead Man's Shoes	2004	NaN	90 min	Crime, Drama, Thriller	
943	944	Batoru rowaiaru	2000	NaN	114 min	Action, Adventure, Drama	
997	998	Lifeboat	1944	NaN	97 min	Drama, War	

	Unnamed: 0	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rati
998	999	The 39 Steps	1935	NaN	86 min	Crime, Mystery, Thriller	

101 rows × 16 columns

In [ ]:
```
df_tratado = df.drop(columns=['Certificate', 'Meta_score', 'Gross'])
```

In [ ]:
```
df_tratado.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Unnamed: 0     999 non-null    int64  
 1   Series_Title   999 non-null    object 
 2   Released_Year  999 non-null    object 
 3   Runtime        999 non-null    object 
 4   Genre          999 non-null    object 
 5   IMDB_Rating    999 non-null    float64
 6   Overview       999 non-null    object 
 7   Director       999 non-null    object 
 8   Star1          999 non-null    object 
 9   Star2          999 non-null    object 
 10  Star3          999 non-null    object 
 11  Star4          999 non-null    object 
 12  No_of_Votes    999 non-null    int64  
dtypes: float64(1), int64(2), object(10)
memory usage: 101.6+ KB
```

In [ ]:
```
#alterar data apollo 13
df_tratado.loc[df_tratado['Series_Title'] == 'Apollo 13', 'Released_Year'] = 199
```

In [ ]:
```
print(df_tratado[df_tratado['Series_Title'] == 'Apollo 13'])
```

```
     Unnamed: 0 Series_Title Released_Year  Runtime  \
965         966    Apollo 13          1995  140 min   

                        Genre  IMDB_Rating  \
965  Adventure, Drama, History          7.6   

                                              Overview    Director      Star1  \
965  NASA must devise a strategy to return Apollo 1...  Ron Howard  Tom Hanks   

           Star2        Star3        Star4  No_of_Votes  
965  Bill Paxton  Kevin Bacon  Gary Sinise       269197  
```

In [ ]:
```
#alterar tipo de dados
df_tratado['Released_Year'] = pd.to_numeric(df_tratado['Released_Year'])
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 16 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Unnamed: 0     999 non-null    int64  
 1   Series_Title   999 non-null    object 
 2   Released_Year  999 non-null    object 
 3   Certificate    898 non-null    object 
 4   Runtime        999 non-null    object 
 5   Genre          999 non-null    object 
 6   IMDB_Rating    999 non-null    float64
 7   Overview       999 non-null    object 
 8   Meta_score     842 non-null    float64
 9   Director       999 non-null    object 
 10  Star1          999 non-null    object 
 11  Star2          999 non-null    object 
 12  Star3          999 non-null    object 
 13  Star4          999 non-null    object 
 14  No_of_Votes    999 non-null    int64  
 15  Gross          830 non-null    object 
dtypes: float64(2), int64(2), object(12)
memory usage: 125.0+ KB
None
```

In [ ]:
```
#descrição dos fatos, podendo ver que os filmes a grande maioria são entre os an
print(df_tratado.describe())
```

```
       Unnamed: 0  Released_Year  IMDB_Rating   No_of_Votes
count  999.000000     999.000000   999.000000  9.990000e+02
mean   500.000000    1991.218218     7.947948  2.716214e+05
std    288.530761      23.297166     0.272290  3.209126e+05
min      1.000000    1920.000000     7.600000  2.508800e+04
25%    250.500000    1976.000000     7.700000  5.547150e+04
50%    500.000000    1999.000000     7.900000  1.383560e+05
75%    749.500000    2009.000000     8.100000  3.731675e+05
max    999.000000    2020.000000     9.200000  2.303232e+06
```

In [ ]:
```
print(df_tratado.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Unnamed: 0     999 non-null    int64  
 1   Series_Title   999 non-null    object 
 2   Released_Year  999 non-null    int64  
 3   Runtime        999 non-null    object 
 4   Genre          999 non-null    object 
 5   IMDB_Rating    999 non-null    float64
 6   Overview       999 non-null    object 
 7   Director       999 non-null    object 
 8   Star1          999 non-null    object 
 9   Star2          999 non-null    object 
 10  Star3          999 non-null    object 
 11  Star4          999 non-null    object 
 12  No_of_Votes    999 non-null    int64  
dtypes: float64(1), int64(3), object(9)
memory usage: 101.6+ KB
None
```

In [ ]:
```
#verificando o tipo de dato de Runtime
print(df_tratado['Runtime'].dtype)
```

```
object
```

In [ ]:
```
print(df_tratado['Runtime'].head())
```

```
0    175 min
1    152 min
2    202 min
3     96 min
4    201 min
Name: Runtime, dtype: object
```

In [ ]:
```
#tirando o "min" do runtime
df_tratado['Runtime'] = df_tratado['Runtime'].str.extract('(\d+)').astype(int)
```

In [ ]:
```
#verificando dados
print(df_tratado['Runtime'].head())
```

```
0    175
1    152
2    202
3     96
4    201
Name: Runtime, dtype: int32
```

In [ ]:
```
#passando para tipo inteiro e verificando
df_tratado['Runtime'] = pd.to_numeric(df_tratado['Runtime'])
print(df_tratado['Runtime'].dtype)
```

```
int32
```

In [ ]:
```
#tipo de dados todos mudados, agora vamos analisar analisar
print(df_tratado.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Unnamed: 0     999 non-null    int64  
 1   Series_Title   999 non-null    object 
 2   Released_Year  999 non-null    int64  
 3   Runtime        999 non-null    int32  
 4   Genre          999 non-null    object 
 5   IMDB_Rating    999 non-null    float64
 6   Overview       999 non-null    object 
 7   Director       999 non-null    object 
 8   Star1          999 non-null    object 
 9   Star2          999 non-null    object 
 10  Star3          999 non-null    object 
 11  Star4          999 non-null    object 
 12  No_of_Votes    999 non-null    int64  
dtypes: float64(1), int32(1), int64(3), object(8)
memory usage: 97.7+ KB
None
```

In [ ]:
```
# buscar a df_tratado por ordem de melhor rating
df_decrescente_imdb = df_tratado.sort_values(by='IMDB_Rating', ascending=False)
print(df_decrescente_imdb.head(30))
```

	Unnamed: 0	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating
0	1	The Godfather	1972	175	Crime, Drama	9.2
2	3	The Godfather: Part II	1974	202	Crime, Drama	9.0
3	4	12 Angry Men	1957	96	Crime, Drama	9.0
1	2	The Dark Knight	2008	152	Action, Crime, Drama	9.0
4	5	The Lord of the Rings: The Return of the King	2003	201	Action, Adventure, Drama	8.9
5	6	Pulp Fiction	1994	154	Crime, Drama	8.9
6	7	Schindler's List	1993	195	Biography, Drama, History	8.9
7	8	Inception	2010	148	Action, Adventure, Sci-Fi	8.8
8	9	Fight Club	1999	139	Drama	8.8
9	10	The Lord of the Rings: The Fellowship of the Ring	2001	178	Action, Adventure, Drama	8.8
10	11	Forrest Gump	1994	142	Drama, Romance	8.8
11	12	Il buono, il brutto, il cattivo	1966	161	Western	8.8
15	16	Star Wars: Episode V - The Empire Strikes Back	1980	124	Action, Adventure, Fantasy	8.7
16	17	One Flew Over the Cuckoo's Nest	1975	133	Drama	8.7
12	13	The Lord of the Rings: The Two Towers	2002	179	Action, Adventure, Drama	8.7
14	15	Goodfellas	1990	146	Biography, Crime, Drama	8.7
13	14	The Matrix	1999	136	Action, Sci-Fi	8.7
25	26	La vita è bella	1997	116	Comedy, Drama, Romance	8.6
31	32	It's a Wonderful Life	1946	130	Drama, Family, Fantasy	8.6
30	31	Shichinin no samurai	1954	207	Action, Adventure, Drama	8.6
29	30	Seppuku	1962	133	Action, Drama, Mystery	8.6
28	29	Star Wars	1977	121	Action, Adventure, Fantasy	8.6
27	28	The Silence of the Lambs	1991	118	Crime, Drama, Thriller	8.6
26	27	Se7en	1995	127	Crime, Drama, Mystery	8.6
21	22	Cidade de Deus	2002	130	Crime, Drama	8.6
24	25	The Green Mile	1999	189	Crime, Drama, Fantasy	8.6
22	23	Sen to Chihiro no kamikakushi	2001	125	Animation, Adventure, Family	8.6
20	21	Interstellar				
19	20	Soorarai Pottru				
18	19	Gisaengchung				

20	2014	169	Adventure, Drama, Sci-Fi	8.6
19	2020	153	Drama	8.6
18	2019	132	Comedy, Drama, Thriller	8.6

	Overview	Director
0	An organized crime dynasty's aging patriarch t...	Francis Ford Coppola
2	The early life and career of Vito Corleone in ...	Francis Ford Coppola
3	A jury holdout attempts to prevent a miscarria...	Sidney Lumet
1	When the menace known as the Joker wreaks havo...	Christopher Nolan
4	Gandalf and Aragorn lead the World of Men agai...	Peter Jackson
5	The lives of two mob hitmen, a boxer, a gangst...	Quentin Tarantino
6	In German-occupied Poland during World War II,...	Steven Spielberg
7	A thief who steals corporate secrets through t...	Christopher Nolan
8	An insomniac office worker and a devil-may-car...	David Fincher
9	A meek Hobbit from the Shire and eight compani...	Peter Jackson
10	The presidencies of Kennedy and Johnson, the e...	Robert Zemeckis
11	A bounty hunting scam joins two men in an unea...	Sergio Leone
15	After the Rebels are brutally overpowered by t...	Irvin Kershner
16	A criminal pleads insanity and is admitted to ...	Milos Forman
12	While Frodo and Sam edge closer to Mordor with...	Peter Jackson
14	The story of Henry Hill and his life in the mo...	Martin Scorsese
13	When a beautiful stranger leads computer hacke...	Lana Wachowski
25	When an open-minded Jewish librarian and his s...	Roberto Benigni
31	An angel is sent from Heaven to help a despera...	Frank Capra
30	A poor village under attack by bandits recruit...	Akira Kurosawa
29	When a ronin requesting seppuku at a feudal lo...	Masaki Kobayashi
28	Luke Skywalker joins forces with a Jedi Knight...	George Lucas
27	A young F.B.I. cadet must receive the help of ...	Jonathan Demme
26	Two detectives, a rookie and a veteran, hunt a...	David Fincher
21	In the slums of Rio, two kids' paths diverge a...	Fernando Meirelles
24	The lives of guards on Death Row are affected ...	Frank Darabont
22	During her family's move to the suburbs, a sul...	Hayao Miyazaki
20	A team of explorers travel through a wormhole ...	Christopher Nolan
19	Nedumaaran Rajangam "Maara" sets out to make t...	Sudha Kongara
18	Greed and class discrimination threaten the ne...	Bong Joon Ho

	Star1	Star2	Star3
0	Marlon Brando	Al Pacino	James Caan
2	Al Pacino	Robert De Niro	Robert Duvall
3	Henry Fonda	Lee J. Cobb	Martin Balsam
1	Christian Bale	Heath Ledger	Aaron Eckhart
4	Elijah Wood	Viggo Mortensen	Ian McKellen
5	John Travolta	Uma Thurman	Samuel L. Jackson
6	Liam Neeson	Ralph Fiennes	Ben Kingsley
7	Leonardo DiCaprio	Joseph Gordon-Levitt	Elliot Page
8	Brad Pitt	Edward Norton	Meat Loaf
9	Elijah Wood	Ian McKellen	Orlando Bloom
10	Tom Hanks	Robin Wright	Gary Sinise
11	Clint Eastwood	Eli Wallach	Lee Van Cleef
15	Mark Hamill	Harrison Ford	Carrie Fisher
16	Jack Nicholson	Louise Fletcher	Michael Berryman
12	Elijah Wood	Ian McKellen	Viggo Mortensen
14	Robert De Niro	Ray Liotta	Joe Pesci
13	Lilly Wachowski	Keanu Reeves	Laurence Fishburne
25	Roberto Benigni	Nicoletta Braschi	Giorgio Cantarini
31	James Stewart	Donna Reed	Lionel Barrymore
30	Toshirô Mifune	Takashi Shimura	Keiko Tsushima
29	Tatsuya Nakadai	Akira Ishihama	Shima Iwashita
28	Mark Hamill	Harrison Ford	Carrie Fisher
27	Jodie Foster	Anthony Hopkins	Lawrence A. Bonney

26	Morgan Freeman	Brad Pitt	Kevin Spacey	
21	Kátia Lund	Alexandre Rodrigues	Leandro Firmino	
24	Tom Hanks	Michael Clarke Duncan	David Morse	
22	Daveigh Chase	Suzanne Pleshette	Miyu Irino	
20	Matthew McConaughey	Anne Hathaway	Jessica Chastain	
19	Suriya	Madhavan	Paresh Rawal	
18	Kang-ho Song	Lee Sun-kyun	Cho Yeo-jeong	

	Star4	No_of_Votes
0	Diane Keaton	1620367
2	Diane Keaton	1129952
3	John Fiedler	689845
1	Michael Caine	2303232
4	Orlando Bloom	1642758
5	Bruce Willis	1826188
6	Caroline Goodall	1213505
7	Ken Watanabe	2067042
8	Zach Grenier	1854740
9	Sean Bean	1661481
10	Sally Field	1809221
11	Aldo Giuffrè	688390
15	Billy Dee Williams	1159315
16	Peter Brocco	918088
12	Orlando Bloom	1485555
14	Lorraine Bracco	1020727
13	Carrie-Anne Moss	1676426
25	Giustino Durano	623629
31	Thomas Mitchell	405801
30	Yukiko Shimazaki	315744
29	Tetsurô Tanba	42004
28	Alec Guinness	1231473
27	Kasi Lemmons	1270197
26	Andrew Kevin Walker	1445096
21	Matheus Nachtergaele	699256
24	Bonnie Hunt	1147794
22	Rumi Hiiragi	651376
20	Mackenzie Foy	1512360
19	Aparna Balamurali	54995
18	Choi Woo-sik	552778

In [ ]:
```
print(df_decrescente_imdb.tail(30))
```

	Unnamed: 0	Series_Title	Released_Year	Runtime	Genre	IMDB_Rating
935	936	Un long dimanche de fiançailles	2004	133	Drama, Mystery, Romance	7.6
907	908	Kick-Ass	2010	117	Action, Comedy, Crime	7.6
934	935	Jeux d'enfants	2003	93	Comedy, Drama, Romance	7.6
933	934	Mysterious Skin	2004	105	Drama	7.6
932	933	Synecdoche, New York	2008	124	Drama	7.6
931	932	Saw	2004	103	Horror, Mystery, Thriller	7.6
930	931	Lord of War	2005	122	Action, Crime, Drama	7.6
929	930	Watchmen	2009	162	Action, Drama, Mystery	7.6
928	929	Match Point	2005	124	Drama, Romance, Thriller	7.6
927	928	300	2006	117	Action, Drama	7.6
926	927	Harry Potter and the Half-Blood Prince	2009	153	Action, Adventure, Family	7.6
925	926	Dead Man's Shoes	2004	90	Crime, Drama, Thriller	7.6
924	925	The Illusionist	2006	110	Drama, Fantasy, Mystery	7.6
923	924	Huo Yuan Jia	2006	104	Action, Biography, Drama	7.6
922	923	La Vie En Rose	2007	140	Biography, Drama, Music	7.6
921	922	Gone Baby Gone	2007	114	Crime, Drama, Mystery	7.6
920	921	Inside Man	2006	129	Crime, Drama, Mystery	7.6
919	920	The Secret of Kells	2009	71	Animation, Adventure, Family	7.6
918	919	Stardust	2007	127	Adventure, Family, Fantasy	7.6
917	918	Eastern Promises	2007	100	Action, Crime, Drama	7.6
916	917	Seven Pounds	2008	123	Drama	7.6
915	916	The Visitor	2007	104	Drama	7.6
914	915	The Blind Side	2009	129	Biography, Drama, Sport	7.6
913	914	Sherlock Holmes	2009	128	Action, Adventure, Mystery	7.6
912	913	Die Welle	2008	107	Drama, Thriller	7.6
911	912	Zombieland	2009	88	Adventure, Comedy, Fantasy	7.6
910	911	La piel que habito	2011	120	Drama, Horror, Thriller	7.6
909	910	Moneyball	2011			
908	909	Celda 211	2009			
998	999	The 39 Steps	1935			

909	133	Biography, Drama, Sport	7.6
908	113	Action, Adventure, Crime	7.6
998	86	Crime, Mystery, Thriller	7.6

	Overview	Director
935	Tells the story of a young woman's relentless ...	Jean-Pierre Jeunet
907	Dave Lizewski is an unnoticed high school stud...	Matthew Vaughn
934	As adults, best friends Julien and Sophie cont...	Yann Samuell
933	A teenage hustler and a young man obsessed wit...	Gregg Araki
932	A theatre director struggles with his work, an...	Charlie Kaufman
931	Two strangers awaken in a room with no recolle...	James Wan
930	An arms dealer confronts the morality of his w...	Andrew Niccol
929	In 1985 where former superheroes exist, the mu...	Zack Snyder
928	At a turning point in his life, a former tenni...	Woody Allen
927	King Leonidas of Sparta and a force of 300 men...	Zack Snyder
926	As Harry Potter begins his sixth year at Hogwa...	David Yates
925	A disaffected soldier returns to his hometown ...	Shane Meadows
924	In turn-of-the-century Vienna, a magician uses...	Neil Burger
923	A biography of Chinese Martial Arts Master Huo...	Ronny Yu
922	Biopic of the iconic French singer Édith Piaf....	Olivier Dahan
921	Two Boston area detectives investigate a littl...	Ben Affleck
920	A police detective, a bank robber, and a high-...	Spike Lee
919	A young boy in a remote medieval outpost under...	Tomm Moore
918	In a countryside town bordering on a magical l...	Matthew Vaughn
917	A teenager who dies during childbirth leaves c...	David Cronenberg
916	A man with a fateful secret embarks on an extr...	Gabriele Muccino
915	A college professor travels to New York City t...	Tom McCarthy
914	The story of Michael Oher, a homeless and trau...	John Lee Hancock
913	Detective Sherlock Holmes and his stalwart par...	Guy Ritchie
912	A high school teacher's experiment to demonstr...	Dennis Gansel
911	A shy student trying to reach his family in Oh...	Ruben Fleischer
910	A brilliant plastic surgeon, haunted by past t...	Pedro Almodóvar
909	Oakland A's general manager Billy Beane's succ...	Bennett Miller
908	The story of two men on different sides of a p...	Daniel Monzón
998	A man in London tries to help a counter-espion...	Alfred Hitchcock

	Star1	Star2	Star3
935	Audrey Tautou	Gaspard Ulliel	Jodie Foster
907	Aaron Taylor-Johnson	Nicolas Cage	Chloë Grace Moretz
934	Guillaume Canet	Marion Cotillard	Thibault Verhaeghe
933	Brady Corbet	Joseph Gordon-Levitt	Elisabeth Shue
932	Philip Seymour Hoffman	Samantha Morton	Michelle Williams
931	Cary Elwes	Leigh Whannell	Danny Glover
930	Nicolas Cage	Ethan Hawke	Jared Leto
929	Jackie Earle Haley	Patrick Wilson	Carla Gugino
928	Scarlett Johansson	Jonathan Rhys Meyers	Emily Mortimer
927	Gerard Butler	Lena Headey	David Wenham
926	Daniel Radcliffe	Emma Watson	Rupert Grint
925	Paddy Considine	Gary Stretch	Toby Kebbell
924	Edward Norton	Jessica Biel	Paul Giamatti
923	Jet Li	Li Sun	Yong Dong
922	Marion Cotillard	Sylvie Testud	Pascal Greggory
921	Morgan Freeman	Ed Harris	Casey Affleck
920	Denzel Washington	Clive Owen	Jodie Foster
919	Nora Twomey	Evan McGuire	Brendan Gleeson
918	Charlie Cox	Claire Danes	Sienna Miller
917	Naomi Watts	Viggo Mortensen	Armin Mueller-Stahl
916	Will Smith	Rosario Dawson	Woody Harrelson
915	Richard Jenkins	Haaz Sleiman	Danai Gurira
914	Quinton Aaron	Sandra Bullock	Tim McGraw

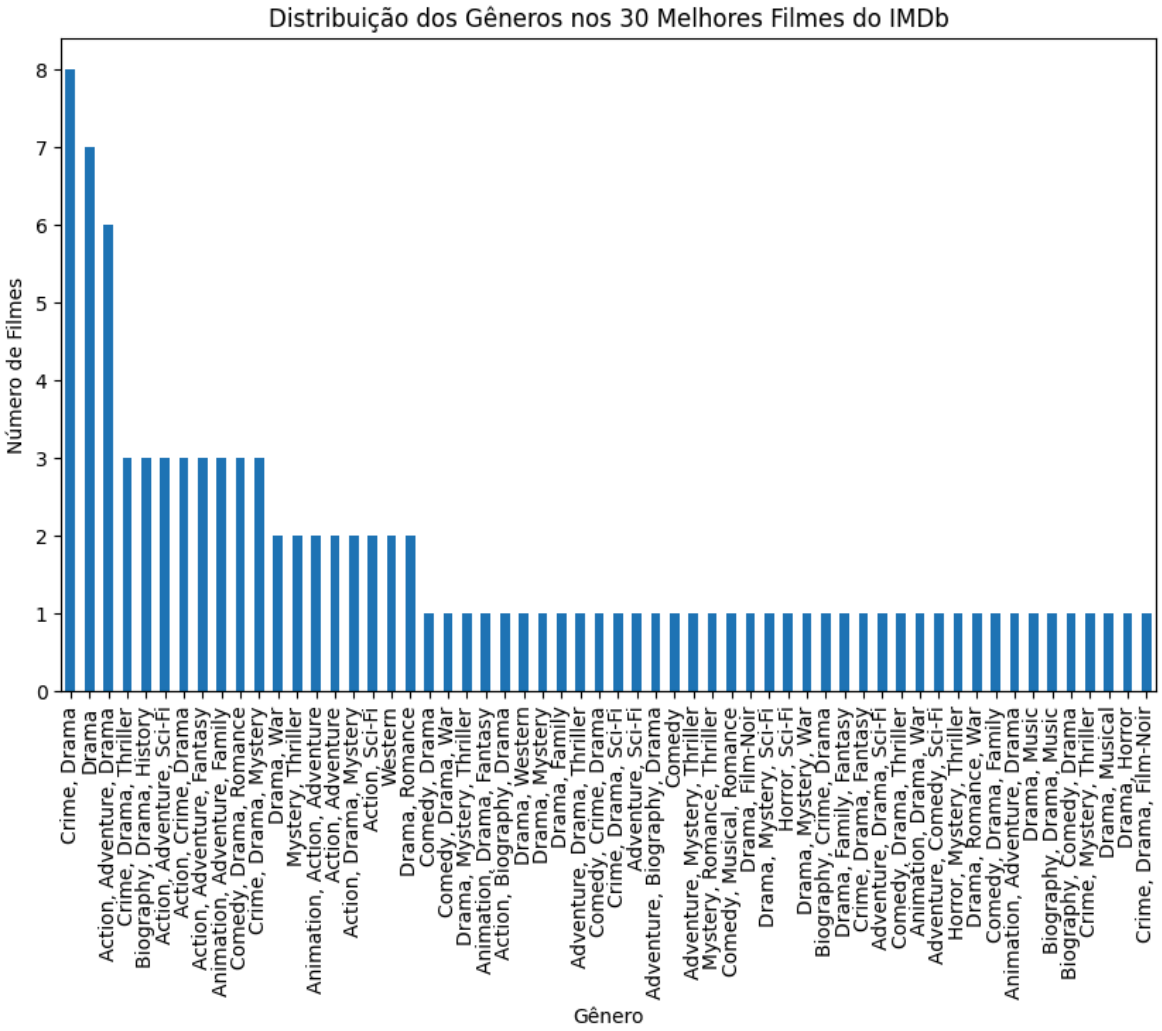
913	Robert Downey Jr.	Jude Law	Rachel McAdams	
912	Jürgen Vogel	Frederick Lau	Max Riemelt	
911	Jesse Eisenberg	Emma Stone	Woody Harrelson	
910	Antonio Banderas	Elena Anaya	Jan Cornet	
909	Brad Pitt	Robin Wright	Jonah Hill	
908	Luis Tosar	Alberto Ammann	Antonio Resines	
998	Robert Donat	Madeleine Carroll	Lucie Mannheim	

	Star4	No_of_Votes
935	Dominique Pinon	70925
907	Garrett M. Brown	524081
934	Joséphine Lebas-Joly	67360
933	Chase Ellison	65939
932	Catherine Keener	83158
931	Ken Leung	379020
930	Bridget Moynahan	294140
929	Malin Akerman	500799
928	Matthew Goode	206294
927	Dominic West	732876
926	Michael Gambon	474827
925	Stuart Wolfenden	49728
924	Rufus Sewell	354728
923	Yun Qu	72863
922	Emmanuelle Seigner	82781
921	Michelle Monaghan	250590
920	Christopher Plummer	339757
919	Mick Lally	31779
918	Ian McKellen	255036
917	Josef Altin	227760
916	Michael Ealy	286770
915	Hiam Abbass	41544
914	Jae Head	293266
913	Mark Strong	583158
912	Jennifer Ulrich	102742
911	Abigail Breslin	520041
910	Marisa Paredes	138959
909	Philip Seymour Hoffman	369529
908	Manuel Morón	63882
998	Godfrey Tearle	51853

In [ ]:
```
import matplotlib.pyplot as plt
```

In [ ]:
```
# Selecionar os 30 melhores filmes
top_30 = df_decrescente_imdb.head(100)

genre_distribution = top_30['Genre'].value_counts()
print(genre_distribution)

# Plotar a distribuição dos gêneros
plt.figure(figsize=(10, 6))
genre_distribution.plot(kind='bar')
plt.xlabel('Gênero')
plt.ylabel('Número de Filmes')
plt.title('Distribuição dos Gêneros nos 30 Melhores Filmes do IMDb')
plt.show()
```

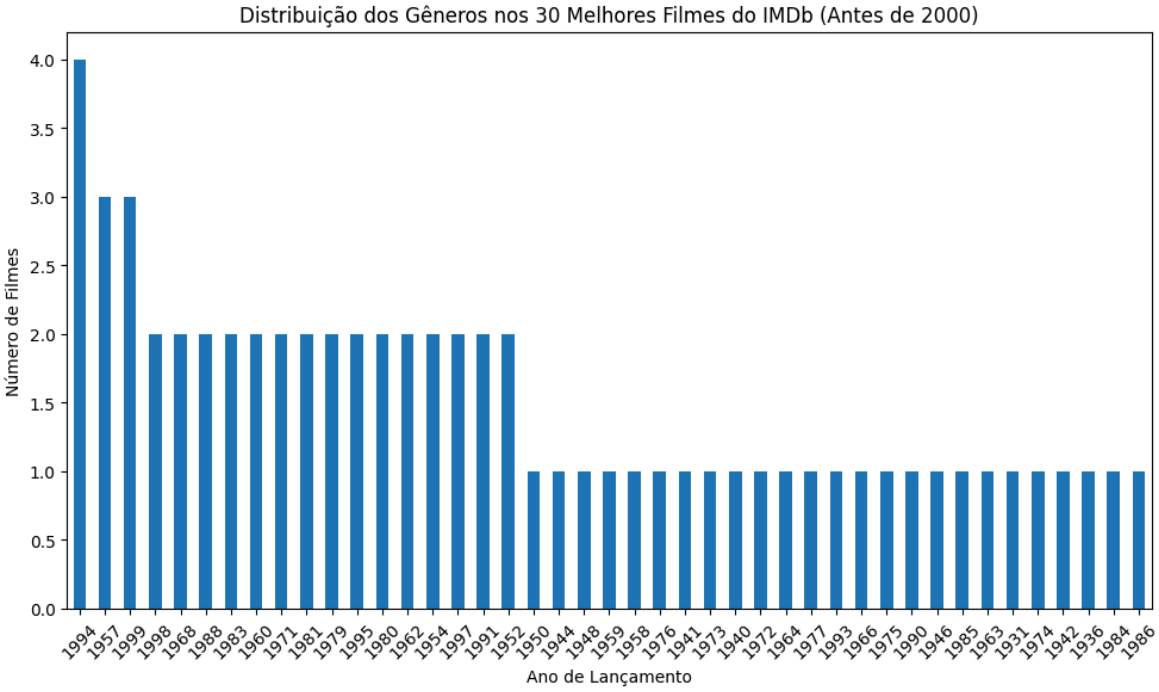
```
Genre
Crime, Drama                     8
Drama                            7
Action, Adventure, Drama         6
Crime, Drama, Thriller           3
Biography, Drama, History        3
Action, Adventure, Sci-Fi        3
Action, Crime, Drama             3
Action, Adventure, Fantasy       3
Animation, Adventure, Family     3
Comedy, Drama, Romance           3
Crime, Drama, Mystery            3
Drama, War                       2
Mystery, Thriller                2
Animation, Action, Adventure     2
Action, Adventure                2
Action, Drama, Mystery           2
Action, Sci-Fi                   2
Western                          2
Drama, Romance                   2
Comedy, Drama                    1
Comedy, Drama, War               1
Drama, Mystery, Thriller         1
Animation, Drama, Fantasy        1
Action, Biography, Drama         1
Drama, Western                   1
Drama, Mystery                   1
Drama, Family                    1
Adventure, Drama, Thriller       1
Comedy, Crime, Drama             1
Crime, Drama, Sci-Fi             1
Adventure, Sci-Fi                1
Adventure, Biography, Drama      1
Comedy                           1
Adventure, Mystery, Thriller     1
Mystery, Romance, Thriller       1
Comedy, Musical, Romance         1
Drama, Film-Noir                 1
Drama, Mystery, Sci-Fi           1
Horror, Sci-Fi                   1
Drama, Mystery, War              1
Biography, Crime, Drama          1
Drama, Family, Fantasy           1
Crime, Drama, Fantasy            1
Adventure, Drama, Sci-Fi         1
Comedy, Drama, Thriller          1
Animation, Drama, War            1
Adventure, Comedy, Sci-Fi        1
Horror, Mystery, Thriller        1
Drama, Romance, War              1
Comedy, Drama, Family            1
Animation, Adventure, Drama      1
Drama, Music                     1
Biography, Drama, Music          1
Biography, Comedy, Drama         1
Crime, Mystery, Thriller         1
Drama, Musical                   1
Drama, Horror                    1
Crime, Drama, Film-Noir          1
Name: count, dtype: int64
```

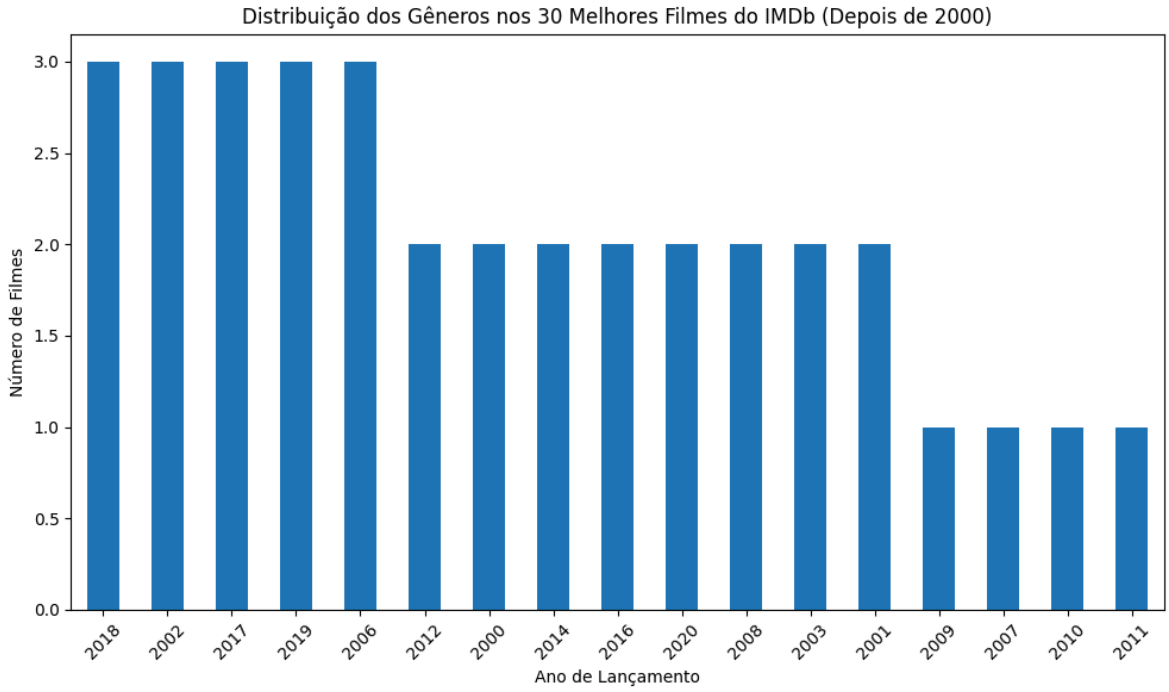


In [ ]:
```
# Buscar o ano dos filmes mais bem avaliados
print(df_decrescente_imdb['Released_Year'].head(30))

# Guarda a lista dos anos dos filmes no top 30 de melhors imdb
released_year_filter = top_30['Released_Year'].value_counts()

#separa usando como parametro o ano 2000
before_2000 = released_year_filter[released_year_filter.index < 2000]
after_2000 = released_year_filter[released_year_filter.index >= 2000]

# Plotar a distribuição de filmes depois dos anos antes de 2000
plt.figure(figsize=(10, 6))
before_2000.plot(kind='bar')
plt.xlabel('Ano de Lançamento')
plt.ylabel('Número de Filmes')
plt.title('Distribuição dos Gêneros nos 30 Melhores Filmes do IMDb (Antes de 200
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Plotar a distribuição de filmes depois dos anos depois de 2000
plt.figure(figsize=(10, 6))
after_2000.plot(kind='bar')
plt.xlabel('Ano de Lançamento')
plt.ylabel('Número de Filmes')
plt.title('Distribuição dos Gêneros nos 30 Melhores Filmes do IMDb (Depois de 20
plt.xticks(rotation=45)
```

```
plt.tight_layout()
plt.show()
```

```
0     1972
2     1974
3     1957
1     2008
4     2003
5     1994
6     1993
7     2010
8     1999
9     2001
10    1994
11    1966
15    1980
16    1975
12    2002
14    1990
13    1999
25    1997
31    1946
30    1954
29    1962
28    1977
27    1991
26    1995
21    2002
24    1999
22    2001
20    2014
19    2020
18    2019
Name: Released_Year, dtype: int64
```





In [ ]:
```
# Filmes com Al Pacino
filmes_al_pacino = df_tratado.loc[df_tratado['IMDB_Rating'] > 9, 'No_of_Votes']
print(filmes_al_pacino)
```

```
0    1620367
Name: No_of_Votes, dtype: int64
```

In [ ]:
```
# Votos do top 30
top_100 = df_decrescente_imdb.head(100)
votos_do_top_100 = top_100['No_of_Votes']
print("A média de votos do top 100 : ")
print(votos_do_top_100.mean())
```

```
A média de votos do top 100 :
729458.14
```

In [ ]:
```
# Votos dos 100 "piores"
top_30_ruim = df_decrescente_imdb.tail(100)
votos_top_30_ruim = top_30_ruim['No_of_Votes']
print("A média de votos do top 100 'piores' : ")
print(votos_top_30_ruim.mean())
```

```
A média de votos do top 100 'piores' :
195368.75
```

In [ ]:
```
# Analise da descrição
top_100.loc[top_100['Overview'].str.contains('crime'), ['Genre', 'Series_Title']
```

Out[ ]:

	Genre	Series_Title
0	Crime, Drama	The Godfather
2	Crime, Drama	The Godfather: Part II
14	Biography, Crime, Drama	Goodfellas
32	Crime, Drama, Thriller	Joker
52	Drama	Capharnaüm

In [ ]:
```
palavras_chave_romance = ['Love', 'love', 'sweetheart', 'relationship', 'affair'
top_100.loc[top_100['Overview'].str.contains('|'.join(palavras_chave_romance)),
```

Out[ ]:

	Genre	Series_Title
10	Drama, Romance	Forrest Gump
14	Biography, Crime, Drama	Goodfellas
18	Comedy, Drama, Thriller	Gisaengchung
44	Drama, Romance	Nuovo Cinema Paradiso
51	Comedy, Drama, Romance	City Lights
49	Drama, Romance, War	Casablanca
41	Action, Crime, Drama	Léon
81	Drama, Film-Noir	Sunset Blvd.
66	Drama, Mystery, Thriller	The Lives of Others
64	Drama, Family	Taare Zameen Par

In [ ]:
```
palavras_chave_acao_ou_drama = ['action', 'adventure', 'battle', 'fight', 'hero'
top_100.loc[top_100['Overview'].str.contains('|'.join(palavras_chave_acao_ou_dra
```

Out[ ]:

	Genre	Series_Title
1	Action, Crime, Drama	The Dark Knight
8	Drama	Fight Club
25	Comedy, Drama, Romance	La vita è bella
30	Action, Adventure, Drama	Shichinin no samurai
28	Action, Adventure, Fantasy	Star Wars
20	Adventure, Drama, Sci-Fi	Interstellar
23	Drama, War	Saving Private Ryan
35	Drama, Mystery, Sci-Fi	The Prestige
32	Crime, Drama, Thriller	Joker
40	Crime, Mystery, Thriller	The Usual Suspects
69	Animation, Action, Adventure	Mononoke-hime
71	Action, Adventure	Raiders of the Lost Ark
74	Horror, Sci-Fi	Alien
79	Drama, War	Paths of Glory
81	Drama, Film-Noir	Sunset Blvd.
58	Action, Adventure, Drama	Avengers: Endgame
54	Action, Crime, Drama	Vikram Vedha
56	Action, Biography, Drama	Dangal
61	Drama, Western	Django Unchained
108	Action, Adventure, Fantasy	Star Wars: Episode VI - Return of the Jedi
110	Crime, Drama	Taxi Driver
115	Adventure, Biography, Drama	Lawrence of Arabia
121	Drama	Ladri di biciclette
105	Action, Adventure, Sci-Fi	Aliens

In [ ]:
```
df_meta_votes_e_imdb = df_tratado[['IMDB_Rating', 'No_of_Votes']]
print(df_meta_votes_e_imdb.corr())
```

```
             IMDB_Rating  No_of_Votes
IMDB_Rating     1.000000     0.479308
No_of_Votes     0.479308     1.000000
```

In [ ]:
```
df_temporario_para_correlacao.corr()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[35], line 1
----> 1 df_temporario_para_correlacao.corr()

NameError: name 'df_temporario_para_correlacao' is not defined
```

In [ ]:
```
df_tratado_tarantio = df_tratado.loc[df['Director'].str.contains('Tarantino'), '
print(df_tratado_tarantio.mean())
```

```
8.175
```

In [ ]:
```
df_tratado_nolan = df_tratado.loc[df['Director'].str.contains('Nolan'), 'IMDB_Ra
print(df_tratado_nolan.mean())
```

```
8.4625
```

In [ ]:
```
df_tratado_coppola = df_tratado.loc[df['Director'].str.contains('Coppola'), 'IMD
print(df_tratado_coppola.mean())
```

```
8.283333333333333
```

In [ ]:
```
df_tratado['Director'].value_counts()
```

Out[ ]:
```
Director
Alfred Hitchcock    14
Steven Spielberg    13
Hayao Miyazaki      11
Akira Kurosawa      10
Martin Scorsese     10
                    ..
Tomas Alfredson      1
Duncan Jones         1
Jacques Audiard      1
Michel Gondry        1
George Stevens       1
Name: count, Length: 548, dtype: int64
```

In [ ]:
```
medias_imdb = {}

for diretor in df_tratado['Director']:
    filmes_diretor = df_tratado[df_tratado['Director'] == diretor]
    media_imdb = filmes_diretor['IMDB_Rating'].mean()
    medias_imdb[diretor] = media_imdb

#for diretor, media in medias_imdb.items():
 #   print(f"Média IMDb Rating - {diretor}: {media:.2f}")

ordenada = pd.DataFrame(list(medias_imdb.items()), columns=['Director', 'IMDB_ra

print(ordenada.sort_values(by='IMDB_rating', ascending=False))
```

```
                Director  IMDB_rating
9         Lana Wachowski          8.7
11        Irvin Kershner          8.7
19      Roberto Benigni           8.6
18        Frank Darabont          8.6
16    Fernando Meirelles          8.6
..                   ...          ...
516       Dennis Gansel           7.6
517    John Lee Hancock           7.6
518    David Cronenberg           7.6
519       Olivier Dahan           7.6
547      George Stevens           7.6

[548 rows x 2 columns]
```

In [ ]:
```
#df_meta_atores_e_imdb = df_tratado[['Director', 'Star1', 'Star2', 'Star3', 'Sta
#print(df_meta_atores_e_imdb.corr())

df_temporario_para_correlacao = df_tratado[['Director', 'IMDB_Rating']]
diretores_unicos = df_temporario_para_correlacao['Director'].unique()
mapeamento_diretores = {diretor: idx + 1 for idx, diretor in enumerate(diretores
df_temporario_para_correlacao['Director'] = df_temporario_para_correlacao['Direc
```

```
C:\Users\Pedro\AppData\Local\Temp\ipykernel_31684\843375126.py:7: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_temporario_para_correlacao['Director'] = df_temporario_para_correlacao['Dire
ctor'].map(mapeamento_diretores)
```

In [ ]:
```
imdb_the_shawshank_redemption = {}
# Tratando para descobrir o IMDB levando em consideração o número de votos, a méd
condicao_No_of_Votes = (df_tratado['No_of_Votes']>2000000) & (df_tratado['No_of_
imdb_the_shawshank_redemption['IMDB_rating_media_votes'] = df_tratado.loc[condic
print(imdb_the_shawshank_redemption)
```

```
{'IMDB_rating_media_votes': 8.9}
```

In [ ]:
```
# Tratando para descobrir o IMDB levando em consideração o diretor, a média
condicao_diretor = (df_tratado['Director']=='Frank Darabont')
imdb_the_shawshank_redemption['IMDB_rating_media_diretor'] = df_tratado.loc[cond
print(imdb_the_shawshank_redemption)
```

```
{'IMDB_rating_media_votes': 8.9, 'IMDB_rating_media_diretor': 8.6}
```

In [ ]:
```
# Tratando para descobrir o IMDB levando em consideração a estrela número 1, a mé
condicao_estrela_1 = (df_tratado['Star1']=='Tim Robbins') | (df_tratado['Star2']
imdb_the_shawshank_redemption['IMDB_rating_media_estrela'] = df_tratado.loc[cond
print(imdb_the_shawshank_redemption)
```

```
{'IMDB_rating_media_votes': 8.9, 'IMDB_rating_media_diretor': 8.6, 'IMDB_rating_m
edia_estrela': 7.733333333333334}
```

In [ ]:
```
# Tratando para descobrir o IMDB levando em consideração a estrela número 2, a mé
condicao_estrela_2 = (df_tratado['Star1']=='Morgan Freeman') | (df_tratado['Star
imdb_the_shawshank_redemption['IMDB_rating_media_estrela_secundaria'] = df_trata
print(imdb_the_shawshank_redemption)
```

```
{'IMDB_rating_media_votes': 8.9, 'IMDB_rating_media_diretor': 8.6, 'IMDB_rating_m
edia_estrela': 7.733333333333334, 'IMDB_rating_media_estrela_secundaria': 8.04000
0000000001}
```

In [ ]:
```
import statistics
```

In [ ]:
```
imdb_ratings_values = list(imdb_the_shawshank_redemption.values())
media_geral = statistics.mean(imdb_ratings_values)
print(f"{media_geral:.2f}")
```

```
8.32
```

In [ ]:
```

```