

Começando o projeto com a EDA, a análise exploratória de dados. A partir disso, vamos poder entender nossa base e ver como os dados estão distribuídos.

```
#importando bibliotecas necessárias para a exploração
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from wordcloud import WordCloud
from collections import Counter
import re
```

```
#Abrir a base de dados
df = pd.read_csv("/content/desafio_indicium_imdb.csv")
```

```
# Entender a estrutura da database
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
```

```
df.describe()
```



	Unnamed: 0	IMDB_Rating	Meta_score	No_of_Votes
count	999.000000	999.000000	842.000000	9.990000e+02
mean	500.000000	7.947948	77.969121	2.716214e+05
std	288.530761	0.272290	12.383257	3.209126e+05
min	1.000000	7.600000	28.000000	2.508800e+04
25%	250.500000	7.700000	70.000000	5.547150e+04
50%	500.000000	7.900000	79.000000	1.383560e+05
75%	749.500000	8.100000	87.000000	3.731675e+05
max	999.000000	9.200000	100.000000	2.303232e+06



```
df.nunique()
```



	0
Unnamed: 0	999
Series_Title	998
Released_Year	100
Certificate	16
Runtime	140
Genre	202
IMDB_Rating	16
Overview	999
Meta_score	63
Director	548
Star1	659
Star2	840
Star3	890
Star4	938
No_of_Votes	998
Gross	822

```
dtype: int64
```

```
print(df.dtypes)
```

```
→ Unnamed: 0      int64
   Series_Title   object
   Released_Year  object
   Certificate     object
   Runtime        object
   Genre          object
   IMDB_Rating    float64
   Overview       object
   Meta_score     float64
   Director       object
   Star1          object
   Star2          object
   Star3          object
   Star4          object
   No_of_Votes    int64
   Gross          object
   dtype: object
```

```
df["Certificate"].value_counts()
```

```
→
```

	count
Certificate	
U	234
A	196
UA	175
R	146
PG-13	43
PG	37
Passed	34
G	12
Approved	11
TV-PG	3
GP	2
TV-14	1
Unrated	1
TV-MA	1
16	1
U/A	1

dtype: int64

Ao explorar a base de dados, podemos perceber que:

- A coluna Unnamed não agrada à base de dados
- A tipificação das colunas não está adequada
- Existem valores nulos nas colunas: Certificate, Meta_score, Gross.
- Os tipos de classificação dos filmes está variado

Para fazer essas modificações, vou trabalhar com uma nova tabela, para manter a original intacta e poder explorar usando uma cópia dela.

```
df_novo = df.copy()
```

```
#Transformar os anos dos filmes em numéricos.
```

```
df_novo['Released_Year'] = pd.to_numeric(df_novo['Released_Year'], errors='coerce')
```

```
#Remover as vírgulas da coluna de faturamento e transformar de object para numérico
```

```
df_novo['Gross'] = df_novo['Gross'].str.replace(',', '', regex=True)
```

```
df_novo['Gross'] = pd.to_numeric(df_novo['Gross'])
```

```
#Remover o "min" dos dados do tempo de filme e transformar em numérico
```


```
df_novo['Runtime'] = df_novo['Runtime'].str.replace(' min', '', regex=True)
```

```
df_novo['Runtime'] = pd.to_numeric(df_novo['Runtime'])
```

```
#Os dados da coluna Gross e Meta_score são importantes e para evitar perder as informações
```

```
df_novo['Gross'].fillna(df_novo['Gross'].mean(), inplace=True)
```

```
df_novo['Meta_score'].fillna(df_novo['Meta_score'].mean(), inplace=True)
```

 /tmp/ipython-input-3214546444.py:2: FutureWarning: A value is trying to be set on a `Categorical` using a tuple. The behavior will change in pandas 3.0. This inplace method will never work because tuples are not hashable.

For example, when doing `'df[col].method(value, inplace=True)'`, try using `'df.method(value, col, inplace=True)'`.

```
df_novo['Gross'].fillna(df_novo['Gross'].mean(), inplace=True)
```


/tmp/ipython-input-3214546444.py:3: FutureWarning: A value is trying to be set on a `Categorical` using a tuple. The behavior will change in pandas 3.0. This inplace method will never work because tuples are not hashable.

For example, when doing `'df[col].method(value, inplace=True)'`, try using `'df.method(value, col, inplace=True)'`.

```
df_novo['Meta_score'].fillna(df_novo['Meta_score'].mean(), inplace=True)
```

```
#Na coluna Certificate vamos substituir os valores nulos por Unrated
```

```
df_novo['Certificate'].fillna('Unrated', inplace=True)
```

 /tmp/ipython-input-2334073304.py:2: FutureWarning: A value is trying to be set on a `Categorical` using a tuple. The behavior will change in pandas 3.0. This inplace method will never work because tuples are not hashable.

For example, when doing `'df[col].method(value, inplace=True)'`, try using `'df.method(value, col, inplace=True)'`.

```
df_novo['Certificate'].fillna('Unrated', inplace=True)
```

```
#Para essa análise vamos excluir as colunas 'Star1', 'Star2', 'Star3', 'Star4', 'Unnamed:
colunas_para_remove = ['Star1', 'Star2', 'Star3', 'Star4', 'Unnamed: 0']
df_novo = df_novo.drop(columns=colunas_para_remove)
```

```
print(df_novo.info())
```

```

↳ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Series_Title          999 non-null    object
1   Released_Year         998 non-null    float64
2   Certificate            999 non-null    object
3   Runtime               999 non-null    int64
4   Genre                 999 non-null    object
5   IMDB_Rating           999 non-null    float64
6   Overview              999 non-null    object
7   Meta_score            999 non-null    float64
8   Director              999 non-null    object
9   No_of_Votes           999 non-null    int64
10  Gross                 999 non-null    float64
dtypes: float64(4), int64(2), object(5)
memory usage: 86.0+ KB
None

```

```
#Os gêneros mais produzidos pelos diretores mais frequentes
```

```
top_diretores = df_novo['Director'].value_counts().nlargest(10).index
df_top_diretores = df_novo[df_novo['Director'].isin(top_diretores)]
```

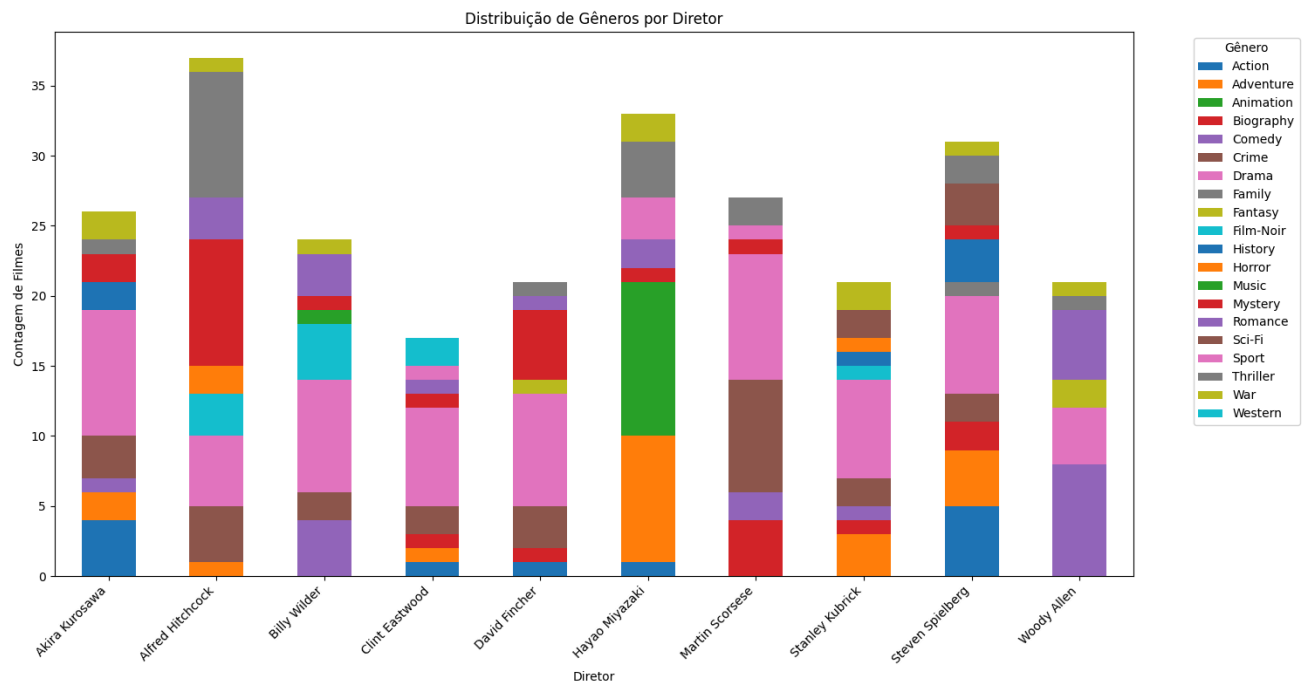
```
df_generos = df_novo.assign(Genre=df_novo['Genre'].str.split(', ')).explode('Genre')
top_10_diretores = df_top_diretores['Director'].value_counts().nlargest(10).index
```

```
contagem_generos_por_diretor = df_generos[df_generos['Director'].isin(top_10_diretores)].
```

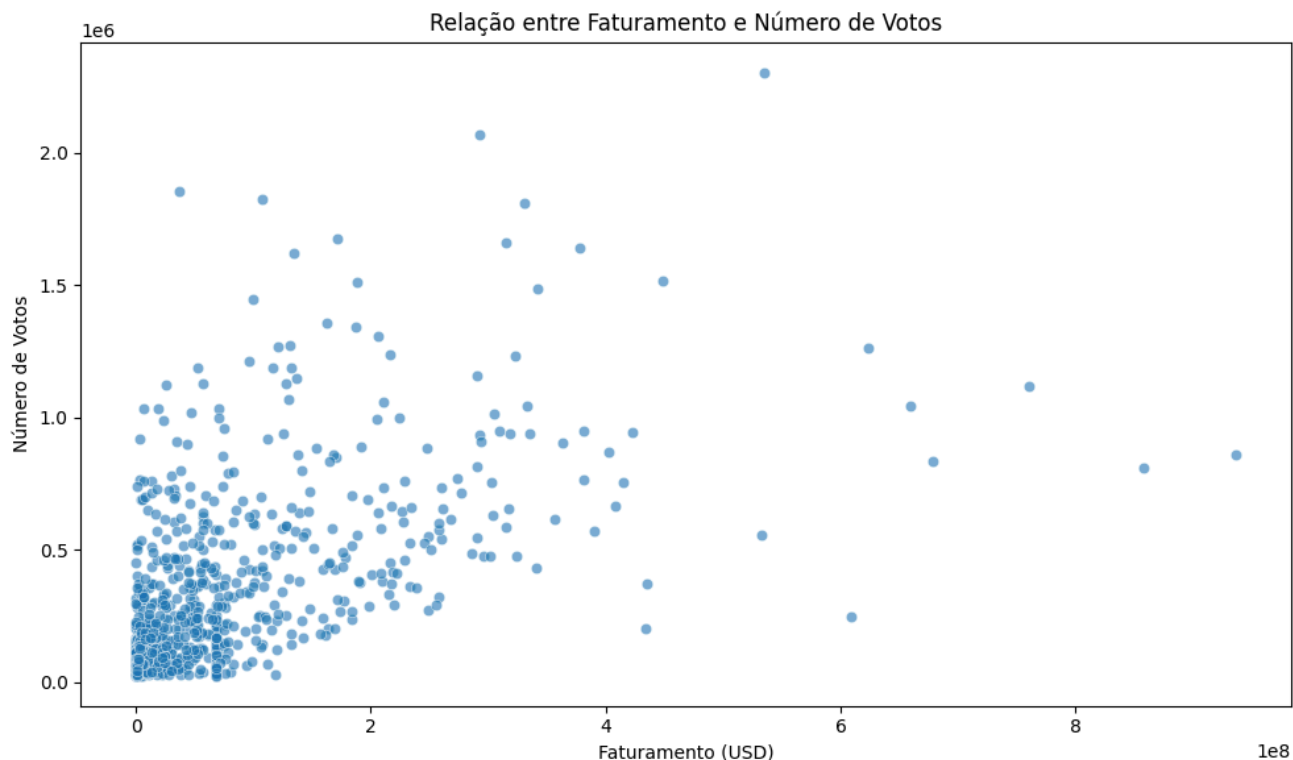
```

contagem_generos_por_diretor.plot(kind='bar', figsize=(15, 8), stacked=True)
plt.title('Distribuição de Gêneros por Diretor')
plt.xlabel('Diretor')
plt.ylabel('Contagem de Filmes')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Gênero', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()

```



```
#Explorar a relação entre o faturamento dos filmes com a quantidade de votos que eles rec
plt.figure(figsize=(10, 6))
sns.scatterplot(x=df_novo['Gross'], y=df_novo['No_of_Votes'], alpha=0.6)
plt.title('Relação entre Faturamento e Número de Votos')
plt.xlabel('Faturamento (USD)')
plt.ylabel('Número de Votos')
plt.tight_layout()
plt.show()
```



```
#Alguns gêneros conseguem alcançar um faturamento mais alto, seja por terem mais filmes d
df_generos = df_novo.assign(Genre=df_novo['Genre'].str.split(', ')).explode('Genre')
```

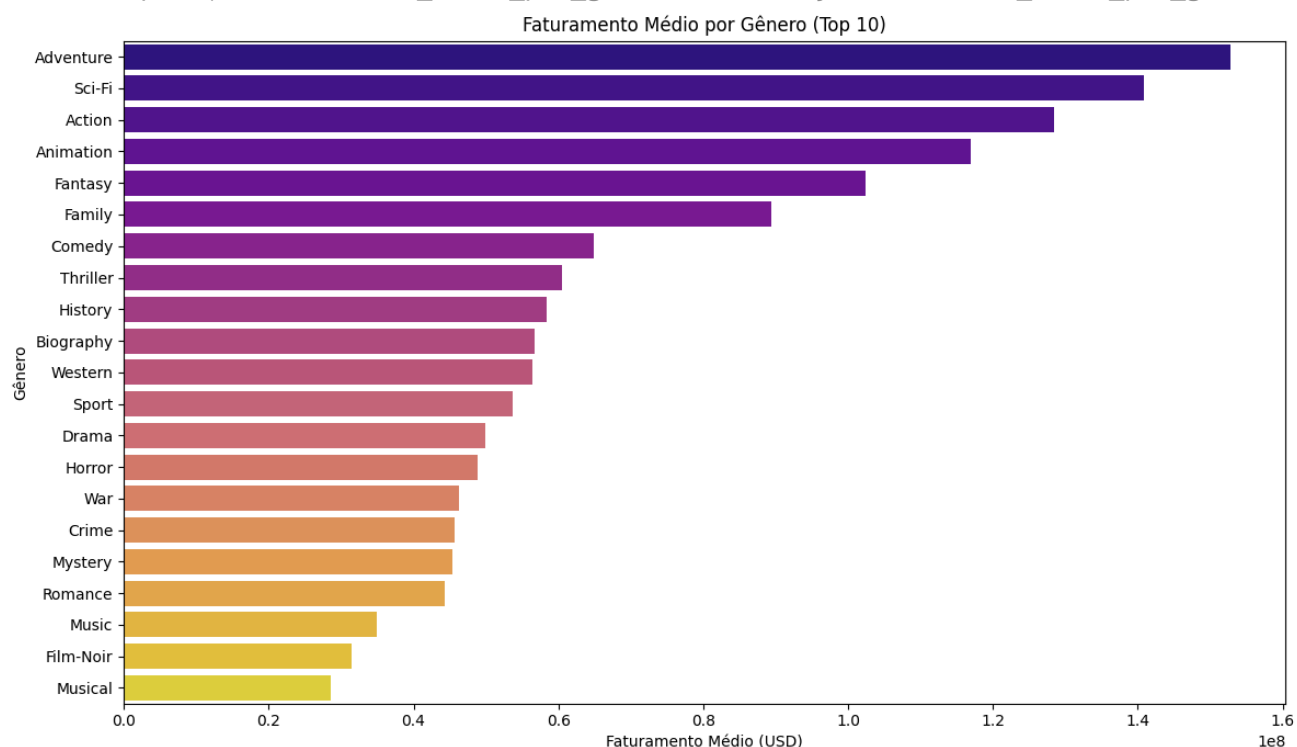
```
faturamento_medio_por_genero = df_generos.groupby('Genre')['Gross'].mean().sort_values(as
```

```
plt.figure(figsize=(12, 7))
sns.barplot(x=faturamento_medio_por_genero.values, y=faturamento_medio_por_genero.index,
plt.title('Faturamento Médio por Gênero (Top 10)')
plt.xlabel('Faturamento Médio (USD)')
plt.ylabel('Gênero')
plt.tight_layout()
plt.show()
```

→ /tmp/ipython-input-3182977225.py:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

```
sns.barplot(x=faturamento_medio_por_genero.values, y=faturamento_medio_por_genero.i
```



```
#Podemos notar que o faturamento não está totalmente ligado ao gênero do filme
df_generos = df_novo.assign(Genre=df_novo['Genre'].str.split(', ')).explode('Genre')
contagem_generos = df_generos['Genre'].value_counts()
```

```
plt.figure(figsize=(12, 8))
sns.barplot(x=contagem_generos.values, y=contagem_generos.index, palette='viridis')
```

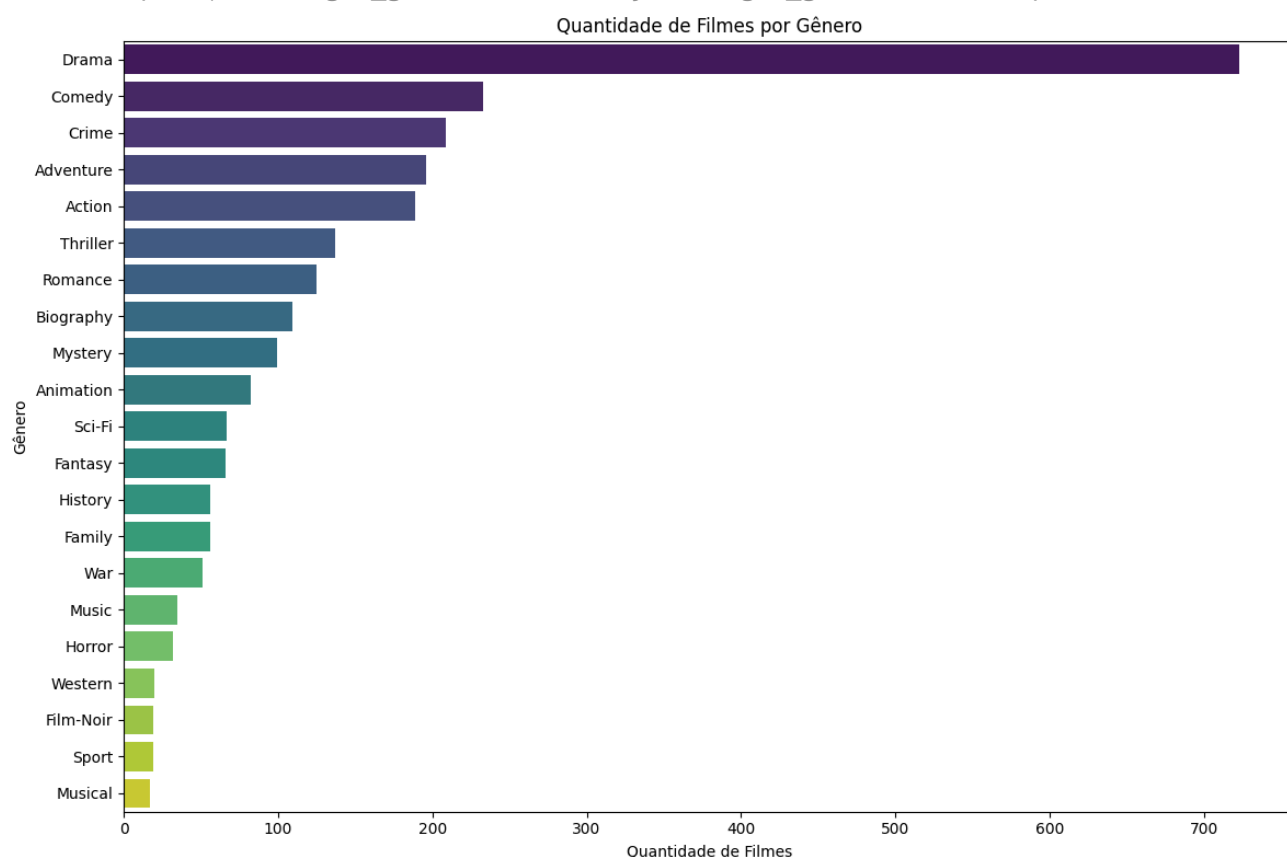
```
plt.title('Quantidade de Filmes por Gênero')
plt.xlabel('Quantidade de Filmes')
plt.ylabel('Gênero')
plt.tight_layout()
```

```
plt.show()
```


↗ /tmp/ipython-input-1616991926.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

```
sns.barplot(x=contagem_generos.values, y=contagem_generos.index, palette='viridis')
```



```
#Analisando se os diretores mais frequentes possuem médias de faturamento acima da média
faturamento_medio_geral = df_novo['Gross'].mean()
```

```
top_diretores = df_novo['Director'].value_counts().nlargest(10).index
df_top_diretores = df_novo[df_novo['Director'].isin(top_diretores)]
```

```
faturamento_medio_por_diretor = df_top_diretores.groupby('Director')['Gross'].mean().sort
```

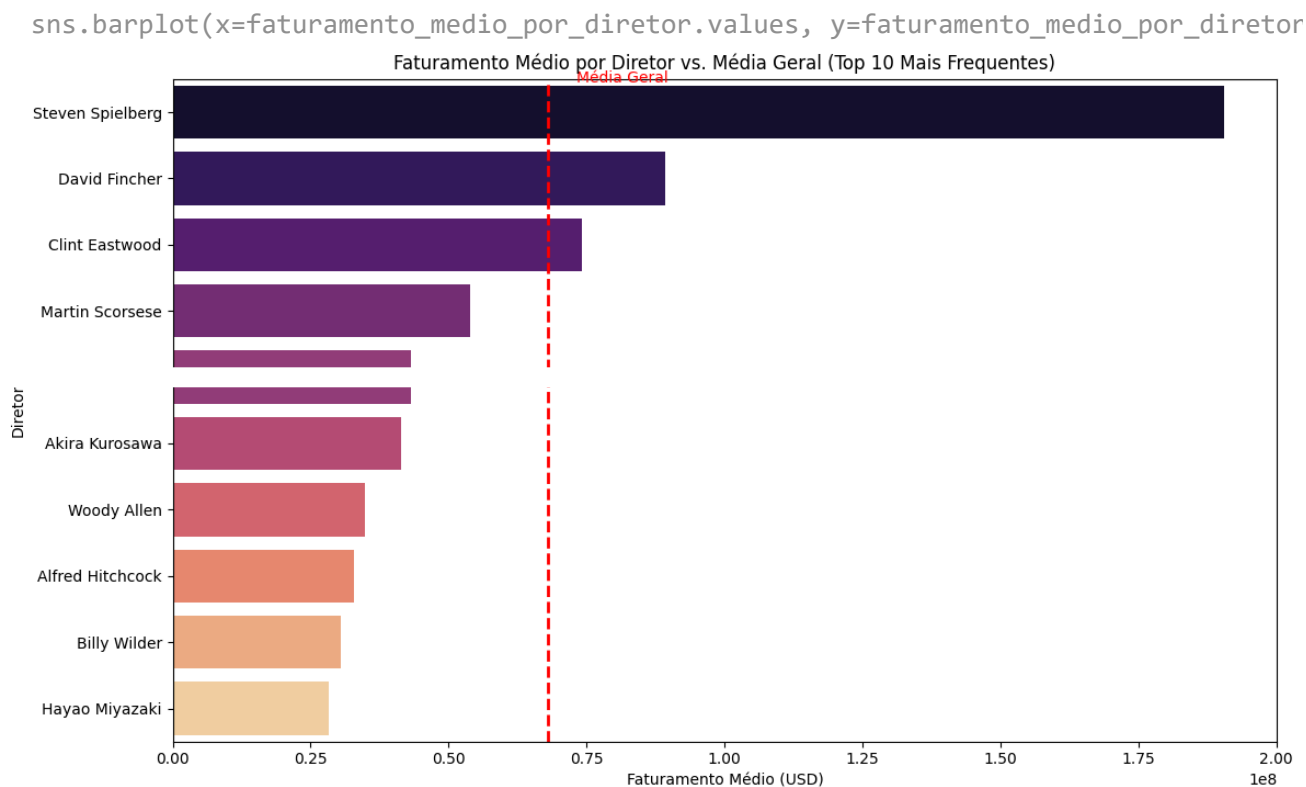
```
plt.figure(figsize=(12, 7))
sns.barplot(x=faturamento_medio_por_diretor.values, y=faturamento_medio_por_diretor.index
```

```
plt.axvline(faturamento_medio_geral, color='red', linestyle='--', linewidth=2)
plt.text(faturamento_medio_geral + 5e6, plt.ylim()[1] * 0.9, 'Média Geral', color='red')
```

```
plt.title('Faturamento Médio por Diretor vs. Média Geral (Top 10 Mais Frequentes)')
plt.xlabel('Faturamento Médio (USD)')
plt.ylabel('Diretor')
plt.tight_layout()
plt.show()
```

→ /tmp/ipython-input-2787503841.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.



```
#Analisando se as notas por diretor mais frequente estão acima da média
media_geral_imdb = df_novo['IMDB_Rating'].mean()
```

```
top_diretores = df_novo['Director'].value_counts().nlargest(10).index
df_top_diretores = df_novo[df_novo['Director'].isin(top_diretores)]
```

```
media_imdb_por_diretor = df_top_diretores.groupby('Director')['IMDB_Rating'].mean().sort_
```

```
plt.figure(figsize=(12, 7))
sns.barplot(x=media_imdb_por_diretor.values, y=media_imdb_por_diretor.index, palette='vir
```

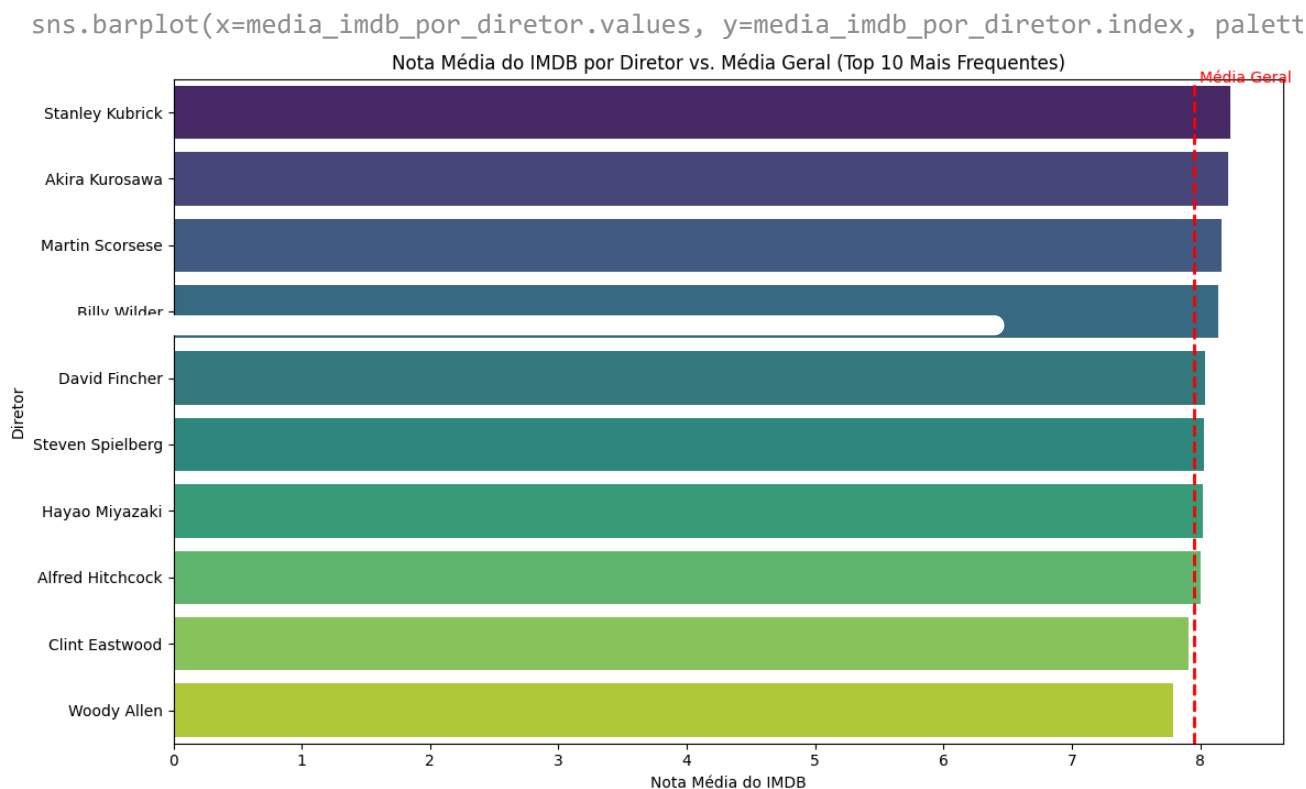
```
plt.axvline(media_geral_imdb, color='red', linestyle='--', linewidth=2)
```

```
plt.text(media_geral_imdb + 0.05, plt.ylim()[1] * 0.9, 'Média Geral', color='red')

plt.title('Nota Média do IMDB por Diretor vs. Média Geral (Top 10 Mais Frequentes)')
plt.xlabel('Nota Média do IMDB')
plt.ylabel('Diretor')
plt.tight_layout()
plt.show()
```

⚠ /tmp/ipython-input-3822952379.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.



A etapa de Análise Exploratória de Dados (EDA) focou no pré-processamento e na formulação de hipóteses.

Para garantir a integridade dos dados, realizei a manipulação em uma cópia do conjunto de dados, denominada `df_novo`. Os valores nulos foram substituídos de forma estratégica para evitar a perda de informações. Além disso, as colunas `Star1`, `Star2`, `Star3`, `Star4` e `Unnamed` foram excluídas, visando simplificar a análise.

Com base nas observações iniciais, foram formuladas as seguintes hipóteses:

- Diretores com maior frequência de filmes no dataset estão
- relacionados a filmes com notas médias altas.
- Os mesmos diretores frequentes também tendem a ter faturamentos médios acima da média geral.
- Alguns gêneros de filmes são mais bem recebidos pelo público, o que se reflete em notas mais altas.
- Existe uma correlação positiva entre o número de votos de um filme e seu faturamento, sugerindo que a popularidade impulsiona o sucesso comercial.

Respondendo as perguntas

✓ Qual filme você recomendaria para uma pessoa que você não conhece?

Por não conhecer a pessoa e não ter indicativo de suas preferências, eu optaria por indicar para ela filmes que foram amplamente assistidos e que ainda sim mantiveram boas avaliações. Assim mostra que foram bem recebidos pela crítica e obtiveram grande sucesso.

Para isso vamos separar os filmes que tiveram notas altas dentre os mais avaliados no IMDB.

```
quartil_votos = df_novo['No_of_Votes'].quantile(0.75)

melhores_filmes = df_novo[
    (df_novo['IMDB_Rating'] >= 8.0) &
    (df_novo['No_of_Votes'] >= quartil_votos)
].sort_values(by=['IMDB_Rating', 'No_of_Votes'], ascending=[False, False])

print("Top 5 Filmes recomendados para uma pessoa que você não conhece:")
print(melhores_filmes[['Series_Title', 'IMDB_Rating', 'No_of_Votes']].head())
```

⇒ Top 5 Filmes recomendados para uma pessoa que você não conhece:

	Series_Title	IMDB_Rating	No_of_Votes
0	The Godfather	9.2	1620367
1	The Dark Knight	9.0	2303232
2	The Godfather: Part II	9.0	1129952
3	12 Angry Men	9.0	689845
5	Pulp Fiction	8.9	1826188

Quais são os principais fatores que estão relacionados com alta expectativa de faturamento de um filme?

Com base nos gráficos, as principais conclusões são:

Número de Votos: Existe uma forte correlação entre o faturamento e o número de votos. Isso significa que filmes com alta popularidade, medidos por votos, tendem a ter um faturamento mais alto.

Gênero: A análise de faturamento médio por gênero mostra que os gêneros de Ação, Aventura e Ficção Científica tendem a gerar faturamentos médios superiores, o que os torna fatores importantes.

Frequencia do Diretor: A comparação do faturamento médio por diretor com a média geral mostrou que diretores mais frequentes tendem a ter faturamentos médios acima da média, o que sugere que a reputação do diretor também é um fator de sucesso.

✓ Quais insights podem ser tirados com a coluna Overview? É possível inferir o gênero do filme a partir dessa coluna?

a) Podemos tirar alguns insights como:

- Quais as palavras mais frequentes nos filmes com maior faturamento
- Quais as palavras mais comuns dentre os gêneros -Quais palavras mais comuns nos filmes com as maiores notas de IBMD

E usando essas palavras e as descrições dos filmes, podemos treinar um modelo para que ele identifique a combinação de palavras melhor aceita pelo público para que ele classifique qual a possível nota do filme no imdb.

b) Sim, é possível, podemos treinar um modelo para entender quais as palavras e as combinações delas são mais comuns por gênero para que ela avalie o genero.

```
#Nesse código, estou separando as palavras mais frequentes dos 5 gêneros com maior quantidade
df_generos = df_novo.assign(Genre=df_novo['Genre'].str.split(', ')).explode('Genre')
generos_comuns = df_generos['Genre'].value_counts().nlargest(5).index
```

```
stopwords = set([
    'a', 'an', 'the', 'and', 'but', 'is', 'in', 'it', 'its', 'to', 'of', 'for', 'with', 'as', 'by', 'at', 'from', 'or', 'he', 'she', 'his', 'her', 'their', 'they', 'who', 'which', 'where', 'when', 'how', 'about', 'just', 'all', 'into', 'be', 'are', 'i', 's', 'over', 'through', 'that', 'this', 'after', 'been', 'before'
])
```

```
for genero in generos_comuns:
    df_filtrado = df_generos[df_generos['Genre'] == genero]
    texto_genero = " ".join(review for review in df_filtrado['Overview'])
```

```
texto_processado = texto_genero.lower()
texto_processado = re.sub(r'[^\w\s]', '', texto_processado)
```

```
palavras = [palavra for palavra in texto_processado.split() if palavra not in stopwords]
```

```
contagem_palavras = Counter(palavras)
lista_frequencia = contagem_palavras.most_common(10)

wordcloud = WordCloud(width=800, height=400, background_color='white').generate(texto

plt.figure(figsize=(10, 8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title(f"Nuvem de Palavras do Gênero: {genero}")
plt.show()

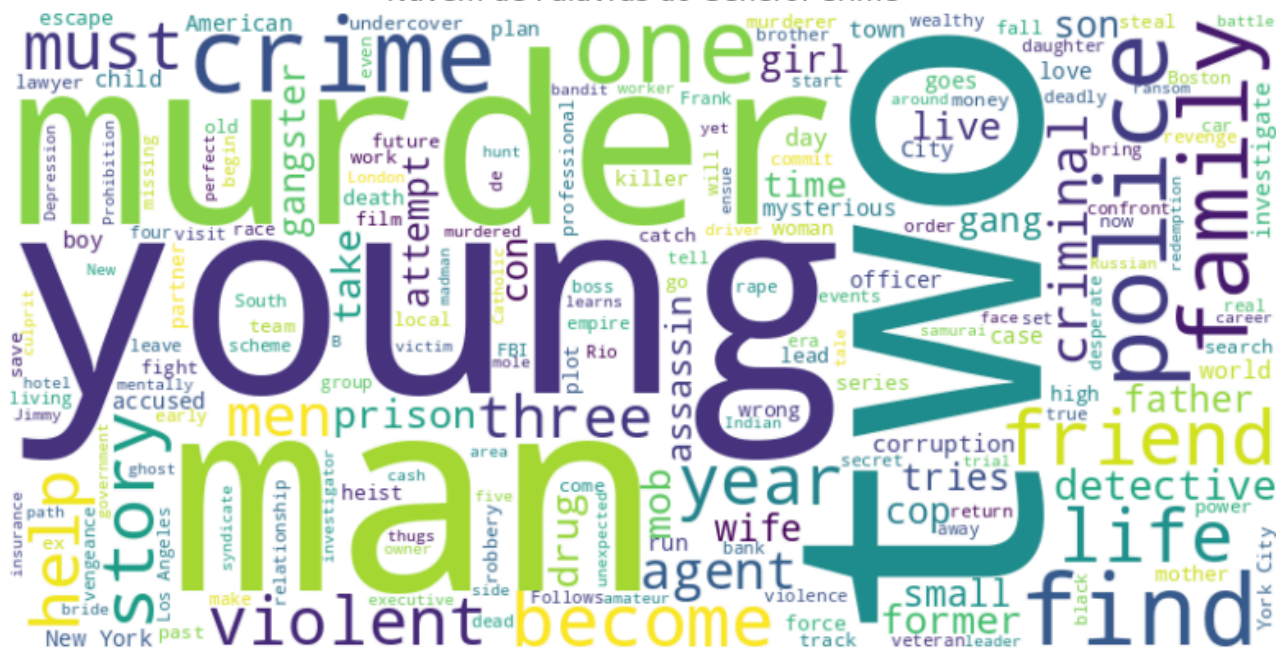
print(f"\n As palavras mais frequentes para o gênero {genero}:")
for palavra, contagem in lista_frequencia:
    print(f"'{palavra}': {contagem}")
```

[illegible]

```
'young': 99
'man': 91
'life': 87
'two': 73
'world': 58
'story': 57
'war': 54
'woman': 53
'new': 49
'him': 48
```

```
'young': 35
'two': 28
'man': 23
'new': 20
'love': 19
'6': 16
```


Nuvem de Palavras do Gênero: Crime



```
'two': 29
'young': 25
'man': 22
'crime': 20
'one': 20
'murder': 20
'police': 19
'him': 18
'family': 16
'them': 16
```

[illegible]

```
'young': 30
```



```
'world': 29
'must': 19
'new': 18
'find': 15
'out': 14
'while': 14
'war': 14
'man': 13
'against': 12
```

Nuvem de Palavras do Gênero: Action



As palavras mais frequentes para o gênero Action:

```
'must': 22
'two': 19
'young': 19
'against': 16
'while': 16
'one': 15
'them': 15
'world': 14
'man': 14
'war': 13
```