

## **Análise de Gêneros Cinematográficos**

### **Integrantes do Grupo**

- Arthur Ferreira - RA00304715
- Manoela Finotti - RA00319067
- Melissa Assis - RA00320501
- Rafael Amaranto - RA00319642

### **Tema e Objetivos do Trabalho**

Primeiramente, foi decidido pelo grupo que o tema seria relacionado a filmes, streamings ou assuntos relacionados a esse ramo. Depois de buscar algumas APIs, foi decidido que seria a “Advanced Movie Search”, do site <https://rapidapi.com/>.

Como forma de organização, foi criado um Trello, para organizar todas as atividades entre os integrantes dos grupos e facilitar a distribuição das tarefas.

Ao longo do estudo da API, o tema do trabalho foi decidido: Análise de Gêneros Cinematográficos.

Dessa forma, o trabalho tem como objetivo analisar os gêneros de filmes que foram disponibilizados pela API, procurando entender quais gêneros tiveram maior alcance, de acordo com algumas variáveis, como o ano, popularidade, média de votos do filme, quantidade de votações, entre outros...

Como etapas do trabalho, foi feito:

- Criação do Trello
- Escolha e tratamento da API
- Análise profunda dos dados
- Banco de dados
- Dashboard
- Criação da API
- Predição

Para acessar o projeto completo, clique [aqui](#).

### **A API**

Como dito anteriormente, a API escolhida foi a “Advanced Movie Search”:

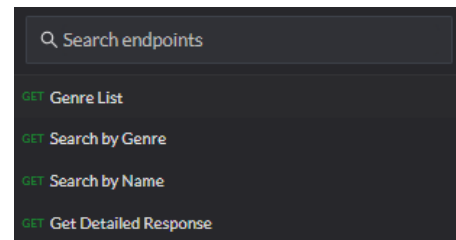
<https://rapidapi.com/jakash1997/api/advanced-movie-search>

Para acessar os dados que eram interessantes ao trabalho, foi necessário escolher quais “endpoints” iriam ser usados.

Foram escolhidos dois:

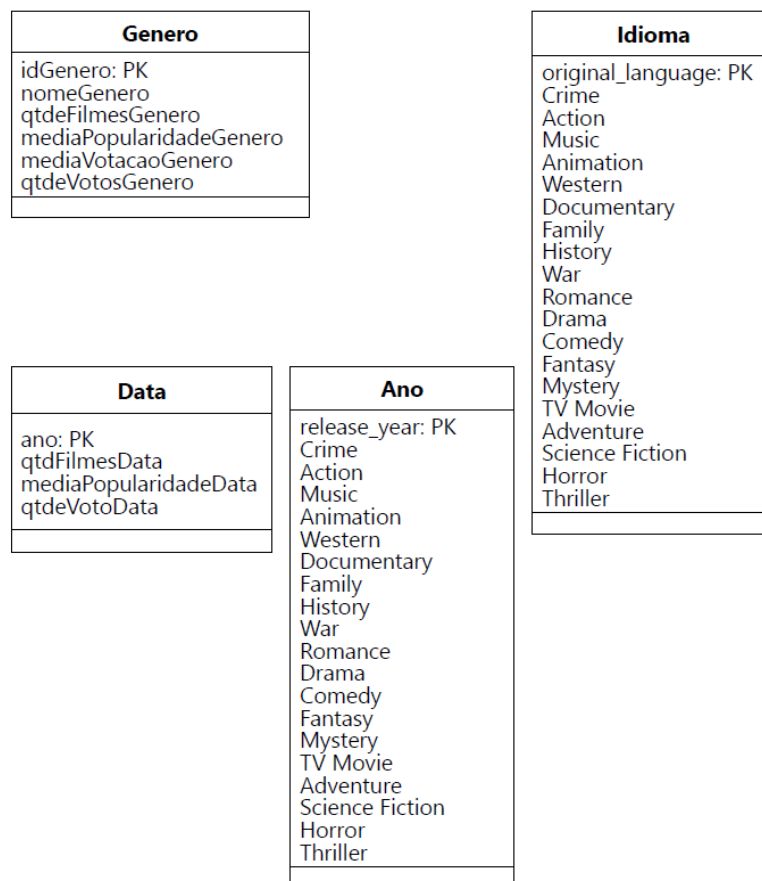
- Genre List
- Search by Genre

Onde foram consumidos e depois transformados em dataframe a partir da linguagem Python.



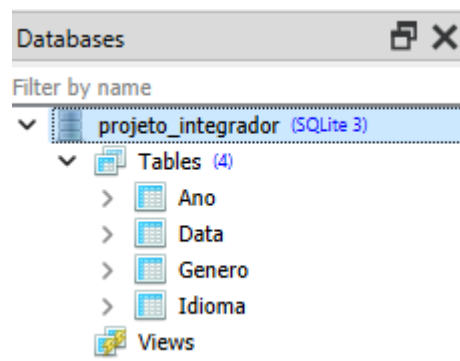
## Banco de Dados

Para iniciar a criação do banco de dados, como forma de organização, foi criado um diagrama simples da estrutura das tabelas:



Na primeira tabela, “Genero”, estão os dados relacionados aos gêneros. Na segunda, “Idioma”, há a quantidade de filmes de acordo com o idioma e com o gênero. Na terceira, “Data”, estão os dados relacionados às datas. Já na quarta, “Ano”, segue a lógica da tabela de idioma, porém de acordo com o ano.

Primeiramente foi criado o banco de dados no SQLite Studio, e as tabelas com as suas respectivas colunas. Abaixo estão os prints das tabelas criadas.



projeto\_integrador Table name: Ano ☐ WITHOUT ROWID ☐ STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated
1	release_year	INTEGER							NULL
2	Crime	INTEGER							NULL
3	Action	INTEGER							NULL
4	Music	INTEGER							NULL
5	Animation	INTEGER							NULL
6	Western	INTEGER							NULL
7	Documentary	INTEGER							NULL
8	Family	INTEGER							NULL
9	History	INTEGER							NULL
10	War	INTEGER							NULL
11	Romance	INTEGER							NULL
12	Drama	INTEGER							NULL

projeto\_integrador Table name: Data ☐ WITHOUT ROWID ☐ STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated
1	ano	INTEGER							NULL
2	qtdFilmesData	INTEGER							NULL
3	mediaPopularidadeData	REAL							NULL
4	qtdeVotoData	INTEGER							NULL

projeto\_integrador Table name: Genero ☐ WITHOUT ROWID ☐ STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated
1	idGenero	INTEGER							NULL
2	nomeGenero	TEXT (20)							NULL
3	qtdeFilmesGenero	INTEGER (5)							NULL
4	mediaPopularidadeGenero	REAL (5, 0)							NULL
5	mediaVotacaoGenero	REAL (5)							NULL
6	qtdeVotosGenero	INTEGER (5)							NULL

projeto\_integra:

☐ WITHOUT ROWID ☐ STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	
1	original_language	TEXT								NULL
2	Crime	INTEGER								NULL
3	Action	INTEGER								NULL
4	Music	INTEGER								NULL
5	Animation	INTEGER								NULL
6	Western	INTEGER								NULL
7	Documentary	INTEGER								NULL
8	Family	INTEGER								NULL
9	History	INTEGER								NULL
10	War	INTEGER								NULL
11	Romance	INTEGER								NULL
12	Drama	INTEGER								NULL

Para a inserção dos dados nas tabelas, de acordo com as queries (para totalização os dados) que foram feitas em Python, foi utilizado um código com duas funções, uma para definir a conexão com o banco de dados, e outra para inserir os dados em questão.

```
[ ] # def ConexaoBanco():
#     caminho = r"projeto_integrador.db"
#     con = None
#     try:
#         con = sqlite3.connect(caminho)
#         print("Conexão estabelecida com sucesso!")
#     except Error as ex:
#         print(ex)
#     return con
# vcon = ConexaoBanco()

# def inserir(conexao, sql):
#     try:
#         c = conexao.cursor()
#         c.execute(sql)
#         conexao.commit()
#         print('Registro Inserido')
#     except Error as ex:
#         print(ex)
```

Como exemplo para a inserção de dados, abaixo está a inserção na “Genero”

▼ Tabela Genero

```
# for index, row in df_genres.iterrows():
#     id_genero = row['id']
#     nome_genero = row['name']
#     filmes_genero = filmes_genres[filmes_genres['genero'] == nome_genero]['qtd_filmes'].iloc[0]
#     popularidade = popularity_genres[popularity_genres['genero'] == nome_genero]['avg_popularidade'].iloc[0]
#     m_voto = avg_vote[avg_vote['genero'] == nome_genero]['avg_voto'].iloc[0]
#     qtd_voto = qty_vote[qty_vote['genero'] == nome_genero]['qtd_voto'].iloc[0]

#     vsql1 = f"INSERT INTO Genero (idGenero, nomeGenero, qtdFilmesGenero, mediaPopularidadeGenero, mediaVotacaoGenero, qtdVotosGenero) VALUES ({id_genero}, '{nome_genero}', {filmes_genero}, {popularidade}, {m_voto}, {qtd_voto})"
#     inserir(vcon, vsql1)
```

> Tabela Data

1 célula oculta

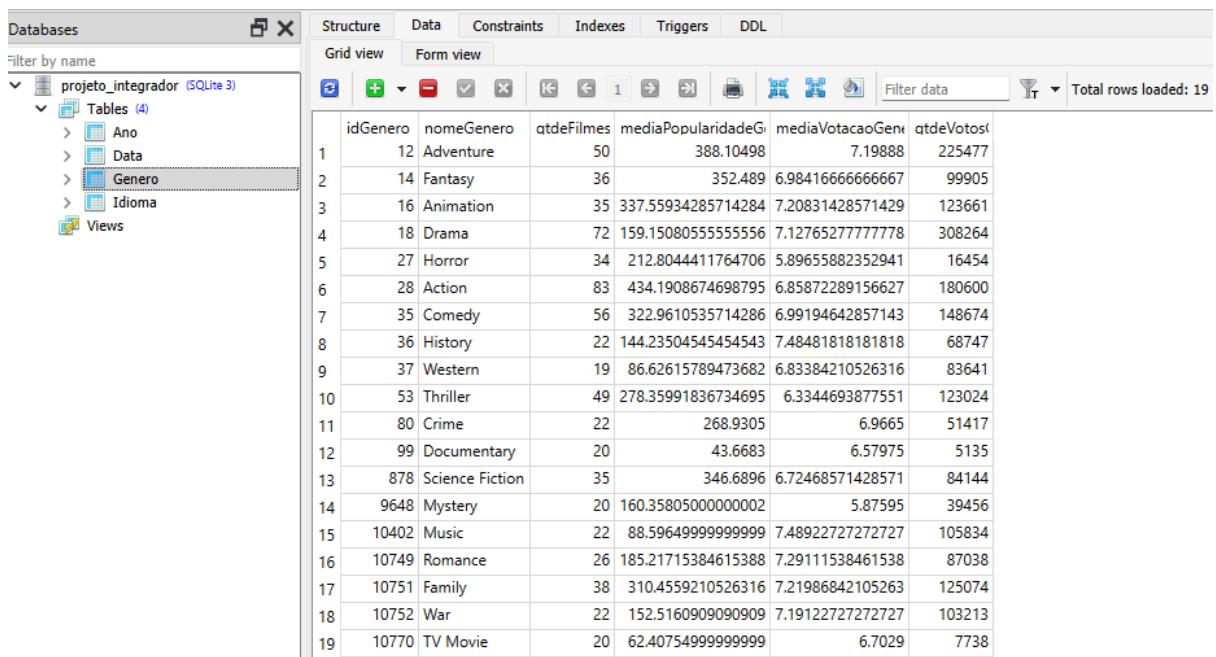
> Tabela Idioma

1 célula oculta

> Tabela Ano

1 célula oculta

Atualizando o SQLite Studio, os dados são transferidos para a tabela:



The screenshot shows the SQLite Studio interface. On the left, the 'Databases' pane shows a project named 'projeto\_integrador (SQLite 3)' containing tables 'Ano', 'Data', 'Genero', and 'Idioma'. The 'Genero' table is selected. The main pane shows the 'Data' tab for the 'Genero' table, displaying 19 rows of data. The columns are: idGenero, nomeGenero, qtdeFilmes, mediaPopularidadeG, mediaVotacaoGen, and qtdeVotosG. The data includes genres like Adventure, Fantasy, Animation, Drama, Horror, Action, Comedy, History, Western, Thriller, Crime, Documentary, Science Fiction, Mystery, Music, Romance, Family, War, and TV Movie.

	idGenero	nomeGenero	qtdeFilmes	mediaPopularidadeG	mediaVotacaoGen	qtdeVotosG
1	12	Adventure	50	388.10498	7.19888	225477
2	14	Fantasy	36	352.489	6.984166666666667	99905
3	16	Animation	35	337.55934285714284	7.20831428571429	123661
4	18	Drama	72	159.15080555555556	7.127652777777778	308264
5	27	Horror	34	212.8044411764706	5.89655882352941	16454
6	28	Action	83	434.1908674698795	6.85872289156627	180600
7	35	Comedy	56	322.9610535714286	6.99194642857143	148674
8	36	History	22	144.23504545454543	7.48481818181818	68747
9	37	Western	19	86.62615789473682	6.83384210526316	83641
10	53	Thriller	49	278.35991836734695	6.3344693877551	123024
11	80	Crime	22	268.9305	6.9665	51417
12	99	Documentary	20	43.6683	6.57975	5135
13	878	Science Fiction	35	346.6896	6.72468571428571	84144
14	9648	Mystery	20	160.35805000000002	5.87595	39456
15	10402	Music	22	88.59649999999999	7.48922727272727	105834
16	10749	Romance	26	185.21715384615388	7.29111538461538	87038
17	10751	Family	38	310.4559210526316	7.21986842105263	125074
18	10752	War	22	152.5160909090909	7.19122727272727	103213
19	10770	TV Movie	20	62.40754999999999	6.7029	7738

## Criação da API

Uma API foi desenvolvida a partir dos dados totalizados nas queries.

Foi criada uma função para obter os dados de cada tabela do banco de dados e, posteriormente, para ser criada cada página da API:

- Página inicial (default), que contém apenas os ids e nomes dos gêneros
- /ano, /data, /genero e /idioma, com os dados de cada tabela do banco de dados.

Como exemplo da página inicial e da tabela ano:

```
app = Flask(__name__)

def getId():
    conn = sqlite3.connect('projeto_integrador.db')
    strSQL = "SELECT idGenero, nomeGenero FROM 'Genero'"
    df_id = pd.read_sql(strSQL, conn)
    conn.close()
    return df_id

def getAno():
    conn = sqlite3.connect('projeto_integrador.db')
    strSQL = "SELECT * FROM 'Ano'"
    df_ano = pd.read_sql(strSQL, conn)
    conn.close()
    return df_ano
```

```
@app.route('/', methods=['GET'])
def index():
    df_id = getId()
    return Response(df_id.to_json(orient = 'records'), mimetype = 'application/json')

@app.route('/ano', methods=['GET'])
def obter_anos():
    df_ano = getAno()
    return Response(df_ano.to_json(orient = 'records'), mimetype = 'application/json')
```

```

{"idGenero":28,"nomeGenero":"Action"}, {"idGenero":12,"nomeGenero":"Adventure"},
{"idGenero":16,"nomeGenero":"Animation"}, {"idGenero":35,"nomeGenero":"Comedy"},
{"idGenero":80,"nomeGenero":"Crime"}, {"idGenero":99,"nomeGenero":"Documentary"},
{"idGenero":18,"nomeGenero":"Drama"}, {"idGenero":10751,"nomeGenero":"Family"},
{"idGenero":14,"nomeGenero":"Fantasy"}, {"idGenero":36,"nomeGenero":"History"},
{"idGenero":27,"nomeGenero":"Horror"}, {"idGenero":10402,"nomeGenero":"Music"},
{"idGenero":9648,"nomeGenero":"Mystery"}, {"idGenero":10749,"nomeGenero":"Romance"},
{"idGenero":878,"nomeGenero":"Science Fiction"}, {"idGenero":10770,"nomeGenero":"TV Movie"},
{"idGenero":53,"nomeGenero":"Thriller"}, {"idGenero":10752,"nomeGenero":"War"},
{"idGenero":37,"nomeGenero":"Western"}]

```

```

{"release_year":1956,"Crime":0,"Action":0,"Music":0,"Animation":0,"Western":0,"Documentary":0,"Family":0,"History":1,"War":0,"Romance":0,"Drama":1,"Comedy":0,"Fantasy":0,"Mystery":0,"TV
Movie":0,"Adventure":0,"Science Fiction":0,"Horror":0,"Thriller":0},
{"release_year":1959,"Crime":0,"Action":1,"Music":0,"Animation":0,"Western":0,"Documentary":0,"Family":0,"History":1,"War":0,"Romance":0,"Drama":1,"Comedy":0,"Fantasy":0,"Mystery":0,"TV
Movie":0,"Adventure":1,"Science Fiction":0,"Horror":0,"Thriller":0},
{"release_year":1962,"Crime":0,"Action":0,"Music":0,"Animation":0,"Western":0,"Documentary":0,"Family":0,"History":0,"War":0,"Romance":1,"Drama":1,"Comedy":1,"Fantasy":0,"Mystery":0,"TV
Movie":0,"Adventure":0,"Science Fiction":0,"Horror":0,"Thriller":0},
{"release_year":1964,"Crime":0,"Action":0,"Music":0,"Animation":0,"Western":1,"Documentary":0,"Family":0,"History":0,"War":0,"Romance":0,"Drama":0,"Comedy":0,"Fantasy":0,"Mystery":0,"TV
Movie":0,"Adventure":0,"Science Fiction":0,"Horror":0,"Thriller":0},
{"release_year":1966,"Crime":0,"Action":0,"Music":0,"Animation":0,"Western":1,"Documentary":0,"Family":0,"History":0,"War":0,"Romance":0,"Drama":0,"Comedy":0,"Fantasy":0,"Mystery":0,"TV
Movie":0,"Adventure":0,"Science Fiction":0,"Horror":0,"Thriller":0},
{"release_year":1995,"Crime":0,"Action":0,"Music":0,"Animation":0,"Western":0,"Documentary":0,"Family":0,"History":0,"War":1,"Romance":0,"Drama":0,"Comedy":0,"Fantasy":0,"Mystery":0,"TV
Movie":0,"Adventure":0,"Science Fiction":0,"Horror":1,"Thriller":0},
{"release_year":1990,"Crime":0,"Action":0,"Music":0,"Animation":0,"Western":1,"Documentary":0,"Family":0,"History":0,"War":0,"Romance":0,"Drama":1,"Comedy":0,"Fantasy":0,"Mystery":0,"TV
Movie":0,"Adventure":1,"Science Fiction":0,"Horror":0,"Thriller":0},
{"release_year":1993,"Crime":0,"Action":1,"Music":0,"Animation":0,"Western":1,"Documentary":0,"Family":0,"History":1,"War":1,"Romance":0,"Drama":1,"Comedy":0,"Fantasy":0,"Mystery":0,"TV
Movie":0,"Adventure":0,"Science Fiction":0,"Horror":0,"Thriller":0},
{"release_year":1995,"Crime":0,"Action":0,"Music":0,"Animation":0,"Western":0,"Documentary":0,"Family":0,"History":0,"War":0,"Romance":0,"Drama":1,"Comedy":0,"Fantasy":0,"Mystery":0,"TV
Movie":1,"Adventure":1,"Science Fiction":0,"Horror":0,"Thriller":0},
{"release_year":1997,"Crime":0,"Action":0,"Music":0,"Animation":0,"Western":0,"Documentary":0,"Family":0,"History":0,"War":0,"Romance":1,"Drama":1,"Comedy":0,"Fantasy":0,"Mystery":0,"TV
Movie":0,"Adventure":0,"Science Fiction":0,"Horror":0,"Thriller":0},
{"release_year":1998,"Crime":0,"Action":0,"Music":0,"Animation":0,"Western":0,"Documentary":0,"Family":0,"History":1,"War":1,"Romance":0,"Drama":1,"Comedy":0,"Fantasy":0,"Mystery":0,"TV
Movie":0,"Adventure":0,"Science Fiction":0,"Horror":0,"Thriller":0},
{"release_year":2000,"Crime":0,"Action":0,"Music":0,"Animation":0,"Western":0,"Documentary":0,"Family":0,"History":0,"War":0,"Romance":0,"Drama":1,"Comedy":0,"Fantasy":0,"Mystery":0,"TV
Movie":1,"Adventure":0,"Science Fiction":0,"Horror":0,"Thriller":0},
{"release_year":2002,"Crime":0,"Action":0,"Music":0,"Animation":1,"Western":1,"Documentary":0,"Family":1,"History":0,"War":0,"Romance":1,"Drama":1,"Comedy":1,"Fantasy":1,"Mystery":0,"TV
Movie":0,"Adventure":2,"Science Fiction":0,"Horror":0,"Thriller":0},
{"release_year":2003,"Crime":0,"Action":1,"Music":1,"Animation":0,"Western":0,"Documentary":0,"Family":0,"History":0,"War":0,"Romance":0,"Drama":0,"Comedy":1,"Fantasy":1,"Mystery":0,"TV
Movie":0,"Adventure":1,"Science Fiction":0,"Horror":0,"Thriller":0},
{"release_year":2004,"Crime":0,"Action":0,"Music":0,"Animation":1,"Western":0,"Documentary":0,"Family":1,"History":0,"War":0,"Romance":0,"Drama":1,"Comedy":1,"Fantasy":1,"Mystery":0,"TV
Movie":1,"Adventure":0,"Science Fiction":1,"Horror":0,"Thriller":0},
{"release_year":2006,"Crime":0,"Action":1,"Music":0,"Animation":1,"Western":0,"Documentary":0,"Family":0,"History":0,"War":1,"Romance":0,"Drama":1,"Comedy":0,"Fantasy":1,"Mystery":0,"TV
Movie":1,"Adventure":0,"Science Fiction":1,"Horror":0,"Thriller":0},
{"release_year":2008,"Crime":0,"Action":1,"Music":0,"Animation":1,"Western":0,"Documentary":0,"Family":1,"History":0,"War":0,"Romance":0,"Drama":0,"Comedy":1,"Fantasy":0,"Mystery":0,"TV
Movie":0,"Adventure":1,"Science Fiction":0,"Horror":0,"Thriller":0},
{"release_year":2009,"Crime":0,"Action":0,"Music":0,"Animation":1,"Western":0,"Documentary":0,"Family":0,"History":0,"War":1,"Romance":0,"Drama":1,"Comedy":0,"Fantasy":1,"Mystery":0,"TV
Movie":0,"Adventure":0,"Science Fiction":0,"Horror":1,"Thriller":1},
{"release_year":2010,"Crime":0,"Action":0,"Music":0,"Animation":0,"Western":0,"Documentary":0,"Family":0,"History":0,"War":0,"Romance":0,"Drama":0,"Comedy":0,"Fantasy":0,"Mystery":0,"TV

```

## Análises dos dados

Primeiramente, após baixar a API e transformá-la em um dataframe, foi iniciada a exploração dos dados.

Inicialmente, foi adicionado o nome dos gêneros (disponíveis em Genre List) ao dataframe com os dados do filme (endpoint Search by Genre), já que esse, não tinham os nomes dos gêneros, apenas os ids.

```

id_to_name = dict(zip(df_genres['id'], df_genres['name']))
id_to_name

```

```

{28: 'Action',
 12: 'Adventure',
 16: 'Animation',
 35: 'Comedy',
 80: 'Crime',
 99: 'Documentary',
 18: 'Drama',
 10751: 'Family',
 14: 'Fantasy',
 36: 'History',
 27: 'Horror',
 10402: 'Music',
 9648: 'Mystery',
 10749: 'Romance',
 878: 'Science Fiction',
 10770: 'TV Movie',
 53: 'Thriller',
 10752: 'War',
 37: 'Western'}

```

```

df['genre_ids'] = df['genre_ids'].apply(ast.literal_eval)

```

```

df['genres_names'] = df['genre_ids'].map(lambda ids: [id_to_name[id] for id in ids])

```

Também foi removido os gêneros duplicados, já que, ao serem adicionados os nomes dos gêneros ao dataframe, como um filme poderia ter mais de um gênero, acabou tendo registros duplicados e idênticos (apenas com a coluna de nome do gênero alterada).

Para uma melhor visualização, anteriormente os países na coluna idioma original estavam com a abreviação, mapeamos, então, para que tivesse o nome do país.

```
[ ] df['original_language'].unique()

array(['en', 'id', 'pt', 'zh', 'es', 'ko', 'it', 'ja', 'th', 'fr', 'hi',
       'da', 'sv', 'tr', 'ru', 'de', 'mn', 'no', 'tl', 'pl', 'fi'],
      dtype=object)
```

```
[ ] map_language = {
    'en': 'English',
    'id': 'Indonesian',
    'pt': 'Portuguese',
    'zh': 'Chinese',
    'es': 'Spanish',
    'ko': 'Korean',
    'it': 'Italian',
    'ja': 'Japanese',
    'th': 'Thai',
    'fr': 'French',
    'hi': 'Hindi',
    'da': 'Danish',
    'sv': 'Swedish',
    'tr': 'Turkish',
    'ru': 'Russian',
    'de': 'German',
    'mn': 'Mongolian',
    'no': 'Norwegian',
    'tl': 'Tagalog',
    'pl': 'Polish',
    'fi': 'Finnish'
}

# Substituindo os códigos de idiomas pelos nomes dos países no DataFrame
df['original_language'] = df['original_language'].map(map_language)
```

Após essas modificações, foram escolhidas apenas as colunas que eram interessantes para a criação do dashboard e predições. Foram removidos filmes com data de lançamento posteriores, já que não teriam dados para analisar.

Por fim, foram criadas variáveis Dummy para os gêneros, que estavam em formato de lista em uma mesma coluna, e passou-se a ter uma coluna para cada gênero, em que caso o valor fosse 0, o filme não estaria naquele gênero, já no caso de 1, sim.

## Gráficos e Dashboard

Inicialmente, foi pensado quais seriam as melhores relações entre as variáveis para a criação de gráficos. Tais gráficos foram criados, a partir das queries feitas.

Após a criação dos gráficos, foi criado um dashboard.



A partir do dashboard, conseguimos observar que os gêneros de Ação, Drama e Terror, em sequência, são os que têm mais quantidade de filmes.



Ação foi o gênero que teve o maior crescimento de filmes ao longo dos anos, de acordo com a nossa API. Já Faroeste, foi o gênero com a maior queda da quantidade de filmes.

Inglês é o idioma que mais tem filmes, principalmente em ação, drama e comédia.

Anterior a 2021, a quantidade de filmes da API é pouca, sendo desproporcional. O ano em que ela tem a maior quantidade de filmes é 2023, com 75 filmes.

Em 2016, houve um grande pico na quantidade de votos, porém o maior foi em 2023, o que faz sentido, já que esse ano tem a maior quantidade de filmes.

A partir de 2020, a média de popularidade tende a crescer, após terem tido picos altos e baixos. Anteriormente a isso, 2008 foi o ano com o maior pico, inclusive, maior que em 2023.

Já em relação aos gêneros mais populares, são: Ação, Aventura e Fantasia. Os gêneros que tiveram mais votos foram: Drama, Aventura e Ação. Documentário e “TV Movie” foram os gêneros com menores votos e popularidade.

A média de votos por gênero foi bem homogênea entre os gêneros.

Como conclusão dessa análise, pode-se dizer que documentários e “TV Movie” são gêneros com menores quantidades de filmes, menor popularidade e com menos quantidades de votos, talvez essa não seja uma boa opção para um filme se tornar popular na indústria cinematográfica. Já os gêneros de ação e drama, são muito bem classificados, sendo uma opção promissora para novos títulos. É um bom momento para os estúdios criarem bons filmes, com gêneros populares, já que estamos em uma época de ascensão em relação a popularidade. Levando em consideração que não estamos nem na metade do ano, e já tem mais que a metade de filmes do que em 2023.

## **Machine Learning**

Levando em consideração a variável target, da média de votos (*vote\_average*), de acordo com o gênero, foram testados modelos de machine learning para a predição de novos filmes, além da análise do dashboard, com os melhores gêneros a se investir.

Primeiramente, foram definidas as variáveis X, tirando os valores que não interessavam para treinar os dados e y, a variável target.

Depois, foi feito o processamento de texto, para a coluna de “overview” dos filmes, removendo números, acentos, pontuações, caracteres especiais, entre outros. Em seguida foi feita a divisão entre treinamento e teste. Pensando na quantidade de dados totais, foram deixados 20% dos dados para teste e 80% para treinamento. Após isso, foi feito o processamento das variáveis categóricas e numéricas.

Não sendo uma variável categórica, mas sim numérica, pensando nos problemas apresentados durante o projeto, foram testados 5 modelos: Linear Regression, Decision Tree, KNN, Random Forest e SVR.

Analizando os resultados e levando em consideração as seguintes métricas de avaliação:

- *Erro Quadrático Médio (Mean Squared Error - MSE)*
  - Mede a média dos quadrados dos erros entre os valores previstos e os reais
  - Quanto menor, melhor
- *Raiz do Erro Quadrático Médio (Root Mean Squared Error - RMSE)*
  - É a raiz quadrada do MSE, fornecendo uma medida do erro médio em unidades da variável dependente
- *Erro Absoluto Médio (Mean Absolute Error - MAE)*
  - Mede a média dos valores absolutos dos erros entre os valores previstos e os valores reais
  - Menos sensível a outliers do que o MSE
  - Um valor menor, indica menor dispersão dos dados, ou seja, mais preciso
- *Coeficiente de Determinação ( $R^2$  - R-squared)*
  - Medida estatística, indicando a proporção da variância dos valores previstos que é explicada pelo modelo
  - Pode variar de 0 a 1; onde 1 indica um ajuste perfeito

Pode-se perceber que o melhor modelo treinado, foi o SVR, tendo um MSE (0.37) mais próximo de zero do que os outros, MAE (0.50) menor e  $R^2$  (0.50) mais próximo de 1. Já o pior modelo, levando em consideração esses mesmos pontos, foi a Decision Tree.

Esses valores não são muito significativos para a nossa análise. Desde que o valor mais próximo de 0 ainda está longe e o mais próximo de 1 também. Porém

como as APIs procuradas apresentavam poucas linhas, os dados que temos para treinar e testar ficam limitados.

### **Como avaliar?**

Os objetivos desse trabalho em questão foram: organização do projeto no Trello, definição do tema, indicado no primeiro tópico deste relatório, consumo da API, criação do banco de dados, criação da API com os dados totalizados, análises dos dados disponíveis, criação de um dashboard e, por fim, predição usando técnicas de Machine Learning.

Nosso grupo conseguiu seguir todos esses objetivos, em tempo agradável e em parceria.