In [ ]:
```
from azure.storage.blob import BlobServiceClient, generate_blob_sas, BlobSasPerm
import pandas as pd
import json
import math
```

Conexão com a plataforma Azure Blob Storage

In [ ]:
```
#enter credentials
account_name = "filmesteste"
account_access_key = "S5wta6QAtuV6yYubHeMSen4mO9CHuP3A0kIYyZmEpEWj6uUugxXzYdrpMu
container_name = "filmesbd"
```

In [ ]:
```
#create a client to interact with blob storage
connect_str = 'DefaultEndpointsProtocol=https;AccountName=' + account_name + ';A
blob_service_client = BlobServiceClient.from_connection_string(connect_str)
```

In [ ]:
```
#use the client to connect to the container
container_client = blob_service_client.get_container_client(container_name)
```

In [ ]:
```
def get_data(file_name):
    blob_client = container_client.get_blob_client(file_name)
    stream_downloader = blob_client.download_blob()
    stream = json.loads(stream_downloader.readall())
    return pd.DataFrame(stream)
```

In [ ]:
```
blob_list = []
for blob_i in container_client.list_blobs():
    blob_list.append(blob_i.name)
print(blob_list)
```

```
['actors.json', 'directors.json', 'movies.json', 'producers.json', 'ratings.jso
n', 'tmdb_5000_movies.json']
```

EXTRAÇÃO DE DADOS

In [ ]:
```
bases = {}
for nome_arquivo in blob_list:
    bases[nome_arquivo] = get_data(nome_arquivo)
```

In [ ]:
```
#redefinindo nome da base de dados utilizadas no trabalho e definidas no catálog
df_movies = bases['movies.json']
df_ratings = bases['ratings.json']
df_box_office = bases['tmdb_5000_movies.json']
#_*_*_**_***_*_**_**_*_***_*_*
df_actors = bases['actors.json']
df_directors = bases['directors.json']
df_producers = bases['producers.json']
```

TRANSFORMAÇÃO DE DADOS

In [ ]:
```
n_movies_without_actors = len(df_movies[df_movies.all_actors.isna()])
n_movies_without_directors = len(df_movies[df_movies.director.isna()])
n_movies_without_producers = len(df_movies[df_movies.producer.isna()])

print('FILMES SEM LISTA DE ATORES: ' + str(n_movies_without_actors))
```

```
print('FILMES SEM LISTA DE DIRETORES: ' + str(n_movies_without_directors))
print('FILMES SEM LISTA DE PRODUTORES: ' + str(n_movies_without_producers))
```

```
FILMES SEM LISTA DE ATORES: 321
FILMES SEM LISTA DE DIRETORES: 267
FILMES SEM LISTA DE PRODUTORES: 930
```

In [ ]:
```
#REMOVE TODOS OS FILMES OS QUAIS NÃO POSSUEM OS ATORES, DIRETORES E PRODUTORES Q

df_movies = df_movies[df_movies.all_actors.notna()]
df_movies = df_movies[df_movies.director.notna()]
df_movies = df_movies[df_movies.producer.notna()]
```

In [ ]:
```
#Redefina o índice do DataFrame e use o padrão.
#inplace = True : Se deve modificar o DataFrame em vez de criar um novo.
#drop = True :This resets the index to the default integer index.
df_movies.reset_index(inplace=True, drop=True)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14238 entries, 0 to 14237
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   title            14238 non-null  object 
 1   year             14238 non-null  int64  
 2   genre            14238 non-null  object 
 3   _id              14238 non-null  object 
 4   director         14238 non-null  object 
 5   all_actors       14238 non-null  object 
 6   gender_percent   14182 non-null  float64
 7   adjusted_budget  5435 non-null   float64
 8   producer         14238 non-null  object 
dtypes: float64(2), int64(1), object(6)
memory usage: 1001.2+ KB
```

CONTABILIZANDO A QUANTIDADE DE ATORES E ATRIZES

In [ ]:
```
df_movies_final = pd.DataFrame()
for index, movie in df_movies.iterrows(): #Iterar sobre linhas do DataFrame como
    male_count = 0
    female_count = 0
    for actor in movie.all_actors:

        if actor['gender'] == 'male':
            male_count += 1
        else:
            female_count +=1

    movie['actors_count'] = male_count
    movie['actresses_count'] = female_count

    df_movies_final = pd.concat([df_movies_final, pd.DataFrame([movie])], ignore
```

CONTABILIZANDO A QUANTIDADE DE PRODUTORES HOMENS E MULHERES

In [ ]:
```
df_movies_final_1 = pd.DataFrame()
for index, movie in df_movies_final.iterrows(): #Iterar sobre linhas do DataFram
```

```
    male_count = 0
    female_count = 0
    for producer in movie.producer:

        if producer['gender'] == 'male':
            male_count += 1
        else:
            female_count +=1

    movie['producer_count_male'] = male_count
    movie['producer_count_female'] = female_count

    df_movies_final_1 = pd.concat([df_movies_final_1, pd.DataFrame([movie])], ig
```

CONTABILIZANDO A QUANTIDADE DE DIRETORES HOMENS E MULHERES

In [ ]:
```
df_movies_final_2 = pd.DataFrame()
for index, movie in df_movies_final_1.iterrows(): #Iterar sobre linhas do DataFr
    male_count = 0
    female_count = 0
    for director in movie.director:

        if director['gender'] == 'male':
            male_count += 1
        else:
            female_count +=1

    movie['director_count'] = male_count
    movie['directress_count'] = female_count

    df_movies_final_2 = pd.concat([df_movies_final_2, pd.DataFrame([movie])], ig
```

REMOVE VARIÁVEIS MULTIVALORADAS

In [ ]:
```
df_movies_final_2.drop(['director', 'producer', 'all_actors'], axis= 1 ,inplace=
#df_movies_final_2.drop(['all_actors'], axis= 1 ,inplace= True )
```

CHECA SE AS VARIÁVEIS CRIADAS DE FATO ESTÃO PRESENTES NO DATASET

In [ ]:
```
df_movies_final_2.columns
```

Out[ ]:
```
Index(['title', 'year', 'genre', '_id', 'gender_percent', 'adjusted_budget',
       'actors_count', 'actresses_count', 'producer_count_male',
       'producer_count_female', 'director_count', 'directress_count'],
      dtype='object')
```

DADOS DE AVALIAÇÕES DO IMDB

In [ ]:
```
df_ratings
```

Out[ ]:

	tconst	averageRating	numVotes
0	tt0000001	5.7	1993
1	tt0000002	5.8	268
2	tt0000003	6.5	1879
3	tt0000004	5.5	177
4	tt0000005	6.2	2662
...	...	...	...
1352356	tt9916730	8.3	10
1352357	tt9916766	7.0	22
1352358	tt9916778	7.2	36
1352359	tt9916840	8.8	6
1352360	tt9916880	8.2	6

1352361 rows × 3 columns

Renomeando a coluna para ter o mesmo nome da coluna de df_movies_final_2

In [ ]:
```
df_ratings.rename(columns={'tconst': '_id'}, inplace=True)
```

Junção dos dados de averageRating e numVotes a df_movies_final_2

In [ ]:
```
df_movies_final_2 = df_movies_final_2.merge(df_ratings, on='_id')
```

In [ ]:
```
df_movies_final_2.head()
```

Out[ ]:

	title	year	genre	_id	gender_percent	adjusted_budg
0	In_Old_Madrid_(1911)	1911	[Comedy, Romance, Short]	tt0001695	50.0	Na
1	Stick_Around_(1925)	1925	[Comedy, Short]	tt0005864	25.0	Na
2	Calling_Dr._Porky_(1940)	1940	[Animation, Comedy, Family, Short]	tt0032298	50.0	Na
3	The_Chewin'_Bruin_(1940)	1940	[Animation, Comedy, Family, Short]	tt0032331	0.0	Na
4	A_House_Divided_(1913/I)	1913	[Comedy, Short]	tt0002983	50.0	Na

TRATAMENTO DE DADOS BOX OFFICE

FORMATA NOME DOS FILMES PARA PADRONIZAR DA MESMA FORMA QUE OS NOMES DE FILMES PRESENTES NO DATAFRAME MOVIES

In [ ]:
```
df_box_office['title'] = df_box_office.title.str.replace(' ', '_')
```

In [ ]:
```
df_box_office.columns
```

Out[ ]:
```
Index(['budget', 'genres', 'homepage', 'id', 'keywords', 'original_language',
       'original_title', 'overview', 'popularity', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'tagline', 'title', 'vote_average',
       'vote_count'],
      dtype='object')
```

REMOVE O ANO DO FINAL DO TITULO DOS FILMES PARA QUE SEJA POSSÍVEL JUNTAR OS DADOS DE AMBOS OS DATAFRAMES PELO TÍTULO POIS O FORMATO DE ID É INCOMPATÍVEL

In [ ]:
```
df_movies_final_2['title'] = df_movies_final_2.title.str.replace("_[\(\[].*?[\)\
```

CAPTURANDO ANO EM RELEASE DATE

In [ ]:
```
df_box_office.head()
```

Out[ ]:

	budget	genres	homepage	id	keyword
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id 146 "name "cultur clash" {"id":
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id 27("name "ocean" {"id": 72("na
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id 47("name "spy" {"id": 81 "name
3	250000000	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	http://www.thedarkknightrises.com/	49026	[{"id 84 "name "c comics" {"id 853,
4	260000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://movies.disney.com/john-carter	49529	[{"id 81 "name "based o novel" {"id":

In [ ]:
```
df_box_office['release_date'] = pd.to_datetime(df_box_office.release_date)
```

In [ ]:
```
df_box_office = df_box_office[df_box_office.release_date.notna()]
```

In [ ]:
```
df_box_office['year'] = df_box_office.release_date.dt.year
df_box_office['year'] = df_box_office.year.astype(int)
```

```
C:\Users\950604\AppData\Local\Temp\ipykernel_20900\4014192577.py:1: SettingWithCo
pyWarning:
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_box_office['year'] = df_box_office.release_date.dt.year
C:\Users\950604\AppData\Local\Temp\ipykernel_20900\4014192577.py:2: SettingWithCo
pyWarning:
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_box_office['year'] = df_box_office.year.astype(int)
```

GARANTE QUE NÃO TERÁ PROBLEMA COM CASE SENSITIVE

In [ ]:
```
df_movies_final_2['title'] = df_movies_final_2.title.str.lower()
df_box_office['title'] = df_box_office.title.str.lower()
```

```
C:\Users\950604\AppData\Local\Temp\ipykernel_20900\1218968412.py:2: SettingWithCo
pyWarning:
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_box_office['title'] = df_box_office.title.str.lower()
```

JUNTA OS DADOS DE BILHETERIA DE BOX OFFICE EM MOVIES

In [ ]:
```
df_movies_consolidated = df_movies_final_2.merge(df_box_office[['title', 'year',
```

In [ ]:
```
df_movies_consolidated = df_movies_consolidated[(df_movies_consolidated.budget >
```

In [ ]:
```
df_movies_consolidated.columns
```

Out[ ]:
```
Index(['title', 'year', 'genre', '_id', 'gender_percent', 'adjusted_budget',
       'actors_count', 'actresses_count', 'producer_count_male',
       'producer_count_female', 'director_count', 'directress_count',
       'averageRating', 'numVotes', 'budget', 'revenue'],
      dtype='object')
```

In [ ]:
```
df_movies_consolidated.shape
```

Out[ ]:
```
(1942, 16)
```

CRIA VARIAVEIS DE PORCENTAGEM COM RECORTE EM GENERO SOCIAL PARA ATORES, DIRETORES E PRODUTORES

In [ ]:
```
df_movies_consolidated['actress_percent'] = df_movies_consolidated.actresses_cou
df_movies_consolidated['actress_percent'] = df_movies_consolidated['actress_perc
df_movies_consolidated['delta_percent'] = df_movies_consolidated.gender_percent
```

In [ ]:
```
df_movies_consolidated['actors_percent'] = df_movies_consolidated.actors_count/(
df_movies_consolidated['actors_percent'] = df_movies_consolidated['actors_percer
```

In [ ]:
```
df_movies_consolidated['women_directing_percent'] = df_movies_consolidated.direc
df_movies_consolidated['women_directing_percent'] = df_movies_consolidated['wome
```

In [ ]:
```
df_movies_consolidated['men_directing_percent'] = df_movies_consolidated.directc
df_movies_consolidated['men_directing_percent'] = df_movies_consolidated['men_di
```

In [ ]:
```
df_movies_consolidated['women_producing_percent'] = df_movies_consolidated.produ
df_movies_consolidated['women_producing_percent'] = df_movies_consolidated['wome
```

In [ ]:
```
df_movies_consolidated['men_producing_percent'] = df_movies_consolidated.produce
df_movies_consolidated['men_producing_percent'] = df_movies_consolidated['men_pr
```

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

```
<class 'pandas.core.frame.DataFrame'>
Index: 1942 entries, 0 to 2481
Data columns (total 23 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   title                    1942 non-null   object 
 1   year                     1942 non-null   int64  
 2   genre                    1942 non-null   object 
 3   _id                      1942 non-null   object 
 4   gender_percent           1942 non-null   float64
 5   adjusted_budget          1854 non-null   float64
 6   actors_count             1942 non-null   int64  
 7   actresses_count          1942 non-null   int64  
 8   producer_count_male      1942 non-null   int64  
 9   producer_count_female    1942 non-null   int64  
 10  director_count           1942 non-null   int64  
 11  directress_count         1942 non-null   int64  
 12  averageRating            1942 non-null   float64
 13  numVotes                 1942 non-null   int64  
 14  budget                   1942 non-null   int64  
 15  revenue                  1942 non-null   int64  
 16  actress_percent          1942 non-null   float64
 17  delta_percent            1942 non-null   float64
 18  actors_percent           1942 non-null   float64
 19  women_directing_percent  1942 non-null   float64
 20  men_directing_percent    1942 non-null   float64
 21  women_producing_percent  1942 non-null   float64
 22  men_producing_percent    1942 non-null   float64
dtypes: float64(10), int64(10), object(3)
memory usage: 364.1+ KB
```

COMPARA SE GENDER_PERCENT É UMA VARIAVEL A SER USADA NO LUGAR DE ACTRESS_PERCENT

In [ ]:
```
df_movies_consolidated[df_movies_consolidated.delta_percent != 0]
```

Out[ ]:

	title	year	genre	_id	gender_percent
11	the_texas_chain_saw_massacre	1974	[Horror, Thriller]	tt0072271	11.0
12	harley_davidson_and_the_marlboro_man	1991	[Action, Crime, Drama, Thriller, Western]	tt0102005	29.0
19	cat_on_a_hot_tin_roof	1958	[Drama]	tt0051459	38.0
24	topaz	1969	[Thriller]	tt0065112	20.0
26	the_exorcist	1973	[Drama, Horror]	tt0070047	36.0
...	...	...	...	...	...
2474	old_dogs	2009	[Comedy, Family]	tt0976238	37.0
2476	yes_man	2008	[Comedy, Romance]	tt1068680	37.0
2479	red	2010	[Action, Comedy, Thriller]	tt1245526	26.0
2480	halloween_ii	2009	[Horror]	tt1311067	28.0
2481	the_next_three_days	2010	[Crime, Drama, Romance, Thriller]	tt1458175	35.0

964 rows × 23 columns

In [ ]:
```
df_movies_consolidated.drop(['_id', 'adjusted_budget', 'delta_percent'], axis= 1
```

In [ ]:
```
df_movies_consolidated
```

Out[ ]:

	title	year	genre	gender_percent	actors_cou
0	snow_white_and_the_seven_dwarfs	1937	[Animation, Family, Fantasy, Musical, Romance]	21.0	1
1	fantasia	1940	[Animation, Family, Fantasy, Music]	14.0	
2	pinocchio	1940	[Adventure, Animation, Drama, Family, Fantasy,...	25.0	
9	alone_with_her	2006	[Crime, Drama, Thriller]	58.0	
10	an_inconvenient_truth	2006	[Documentary]	0.0	
...	...	...	...	...	
2477	the_wrestler	2008	[Drama, Romance, Sport]	19.0	9
2478	i_hope_they_serve_beer_in_hell	2009	[Comedy]	44.0	5
2479	red	2010	[Action, Comedy, Thriller]	26.0	6
2480	halloween_ii	2009	[Horror]	28.0	5
2481	the_next_three_days	2010	[Crime, Drama, Romance, Thriller]	35.0	6

1942 rows × 20 columns

CRIA VARIAVEIS DUMMIES PARA GÊNEROS DE FILMES

In [ ]:
```
df = df_movies_consolidated.explode('genre', ignore_index=True)
```

In [ ]:
```
generos = df.genre.unique()
```

In [ ]:
```
generos
```

Out[ ]:
```
array(['Animation', 'Family', 'Fantasy', 'Musical', 'Romance', 'Music',
       'Adventure', 'Drama', 'Crime', 'Thriller', 'Documentary', 'Horror',
       'Action', 'Western', 'Comedy', 'War', 'Sci-Fi', 'Sport', 'History',
       'Biography', 'Mystery', 'Film-Noir', 'Adult'], dtype=object)
```

In [ ]:
```
df_final = pd.DataFrame()
for index, row in df_movies_consolidated.iterrows():
    for genero in generos:
```

```
        row[genero] = 1 if genero in row.genre else 0
    df_final = pd.concat([df_final, pd.DataFrame([row])])
```

In [ ]:
```
df_final.drop('genre', axis=1, inplace=True)
```

In [ ]:
```
df_final
```

Out[ ]:

	title	year	gender_percent	actors_count	actresses_co
0	snow_white_and_the_seven_dwarfs	1937	21.0	11	
1	fantasia	1940	14.0	6	
2	pinocchio	1940	25.0	9	
9	alone_with_her	2006	58.0	5	
10	an_inconvenient_truth	2006	0.0	4	
...	...	...	...	...	
2477	the_wrestler	2008	19.0	93	
2478	i_hope_they_serve_beer_in_hell	2009	44.0	50	
2479	red	2010	26.0	66	
2480	halloween_ii	2009	28.0	57	
2481	the_next_three_days	2010	35.0	65	

1942 rows × 42 columns

In [ ]:
```
df_final.to_json('df_movies_consolidated.json')
```

In [ ]:
```

```