

(https://databricks.com) Objetivo do trabalho

• objetivo é identificar qual genero tem a melhor nota IMDb e melhor Metascore. Objetivos secundários: qual ano teve a melhor média entre o genero com melhor nota; quais generos tiveram melhores notas em cada ano

Base de dados

 https://www.kaggle.com/datasets/parthdande/imdb-dataset-2024-updated (https://www.kaggle.com/datasets/parthdande/imdb-dataset-2024-updated)

Coleta de Dados

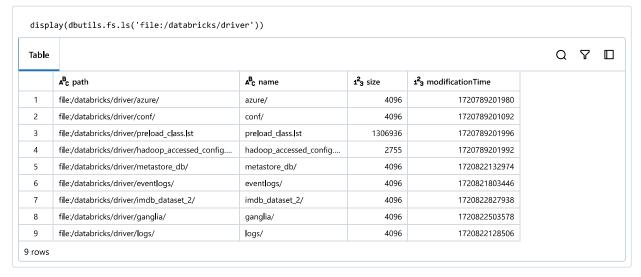
```
!pip install kaggle
Requirement already satisfied: kaggle in /local_disk0/.ephemeral_nfs/envs/pythonEnv-ccde3985-a79a-48d6-93ad-d5476ffefb03/li
b/python3.9/site-packages (1.6.14)
Requirement\ already\ satisfied:\ python-slugify\ in\ /local\_disk0/.ephemeral\_nfs/envs/pythonEnv-ccde3985-a79a-48d6-93ad-d5476ffe
fb03/lib/python3.9/site-packages (from kaggle) (8.0.4)
Requirement already satisfied: python-dateutil in /databricks/python3/lib/python3.9/site-packages (from kaggle) (2.8.2)
Requirement already satisfied: urllib3 in /databricks/python3/lib/python3.9/site-packages (from kaggle) (1.26.9)
Requirement already satisfied: six>=1.10 in /databricks/python3/lib/python3.9/site-packages (from kaggle) (1.16.0)
Requirement already satisfied: bleach in /databricks/python3/lib/python3.9/site-packages (from kaggle) (4.1.0)
Requirement already satisfied: tqdm in /local_disk0/.ephemeral_nfs/envs/pythonEnv-ccde3985-a79a-48d6-93ad-d5476ffefb03/lib/p
ython3.9/site-packages (from kaggle) (4.66.4)
Requirement already satisfied: certifi>=2023.7.22 in /local_disk0/.ephemeral_nfs/envs/pythonEnv-ccde3985-a79a-48d6-93ad-d547
6ffefb03/lib/python3.9/site-packages (from kaggle) (2024.7.4)
Requirement already satisfied: requests in /databricks/python3/lib/python3.9/site-packages (from kaggle) (2.27.1)
Requirement already satisfied: webencodings in /databricks/python3/lib/python3.9/site-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: packaging in /databricks/python3/lib/python3.9/site-packages (from bleach->kaggle) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /databricks/python3/lib/python3.9/site-packages (from packaging->
bleach->kaggle) (3.0.4)
Requirement already satisfied: text-unidecode>=1.3 in /local_disk0/.ephemeral_nfs/envs/pythonEnv-ccde3985-a79a-48d6-93ad-d54
76ffefb03/lib/python3.9/site-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: idna<4.>=2.5 in /databricks/python3/lib/python3.9/site-packages (from requests->kaggle) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in /databricks/python3/lib/python3.9/site-packages (from requests-> `
```

```
import os
from kaggle.api.kaggle_api_extended import KaggleApi

def authenticate_kaggle_api(kaggle_username, kaggle_key) -> KaggleApi:
    os.environ['KAGGLE_USERNAME'] = kaggle_username
    os.environ['KAGGLE_KEY'] = kaggle_key
    api = KaggleApi()
    api.authenticate()
    print("API authenticated.")

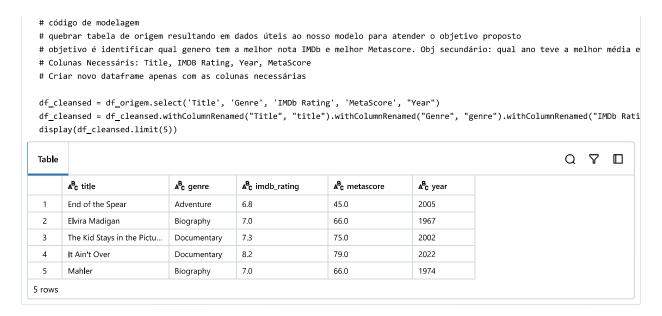
return api
```

```
# kaggle api tokens
      import pandas as pd
      kaggle_username = 'thiagonaves'
      kaggle_key = '94710cfe09512e6494cfbe92e70be222'
                   api = authenticate_kaggle_api(kaggle_username, kaggle_key)
      except e as Exception:
                   print(f'Erro ao autenticar a api do kaggle: {Exception}')
     dataset_name = 'parthdande/imdb-dataset-2024-updated'
     download path = 'imdb dataset 2'
      if not os.path.exists(download_path):
                   os.makedirs(download_path)
      # Baixe o dataset
      api.dataset_download_files(dataset_name, path=download_path, unzip=True)
     print(f"Os arquivos foram baixados e extraídos para: {download_path}")
API authenticated.
Dataset URL: https://www.kaggle.com/datasets/parthdande/imdb-dataset-2024-updated (https://www.kaggle.com/datasets/parthdande/imdb-dataset-2024-updated (https://www.kaggle.com/dataset-2024-updated (https://www.kaggle.com/datas
imdb-dataset-2024-updated)
Os arquivos foram baixados e extraídos para: imdb_dataset_2
```





Modelagem



Análise da Qualidade dos Dados

```
# Transformar imdb_rating e metascore em decimal e year em int. E dropar linahs onde a conversão der errado.
   from pyspark.sql.functions import col, when
   from pyspark.sql.types import DecimalType, IntegerType
   # Define o esquema para as conversões
   decimal_type = DecimalType(4, 1) # Define a precisão e escala. Ajuste conforme necessário.
   integer_type = IntegerType()
   # Converta as colunas para DecimalType e remove linhas inválidas
  df transformed = (
             df cleansed
               .withColumn("imdb_rating", when(col("imdb_rating").rlike("^[0-9]*\\.?[0-9]+$"), col("imdb_rating").cast(decimal_type)).oth
               . with Column ("metascore", when (col("metascore").rlike("^[0-9]*\\.?[0-9]+$"), col("metascore"). cast(decimal\_type)). otherwise (column ("metascore").rlike("^[0-9]*\\.?[0-9]+$"), col("metascore"). cast(decimal_type)). otherwise (column ("metascore").rlike("^[0-9]*\\.?[0-9]*\\.?[0-9]+$"), col("metascore"). cast(decimal_type)). otherwise (column ("metascore").rlike("^[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]*\\.?[0-9]
               . with Column ("year", when (col("year").rlike ("^[0-9]{4}$"), col("year").cast (integer\_type)). otherwise (None)) \\
               .dropna(subset=["imdb_rating", "metascore"])
  display(df_transformed.limit(5))
                                                                                                                                                                                                                                                                                                                      Q T
  Table
                                                                                                                                                                                                                                                                                                                                                 ABc title
                                                                                  <sup>B</sup>c genre
                                                                                                                         .00 imdb_rating
                                                                                                                                                                               .00 metascore
                                                                                                                                                                                                                                 123 year
                   End of the Spear
                                                                                  Adventure
                                                                                                                                                                                                                 45.0
                                                                                                                                                                                                                                                    2005
     1
                                                                                                                                                                  6.8
     2
                   Elvira Madigan
                                                                                  Biography
                                                                                                                                                                  7.0
                                                                                                                                                                                                                 66.0
                                                                                                                                                                                                                                                    1967
                    The Kid Stays in the Pictu...
                                                                                                                                                                  7.3
                                                                                                                                                                                                                 75.0
                                                                                                                                                                                                                                                    2002
     3
                                                                                  Documentary
    4
                   It Ain't Over
                                                                                  Documentary
                                                                                                                                                                  8.2
                                                                                                                                                                                                                 79.0
                                                                                                                                                                                                                                                    2022
     5
                   Mahler
                                                                                   Biography
                                                                                                                                                                   7.0
                                                                                                                                                                                                                 66.0
                                                                                                                                                                                                                                                    1974
5 rows
```

Carga de dados

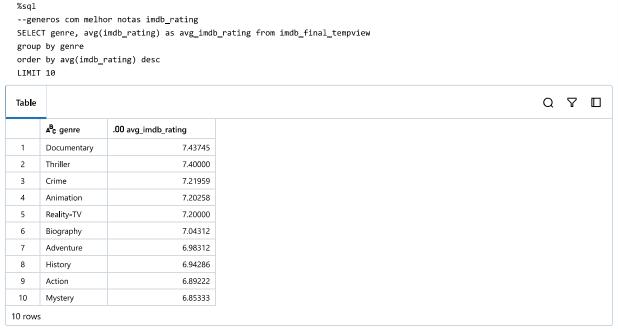
```
# escrever as tabelas ou dados
try:
    permanent_table_name = "mvp_imdb_final_dataset_parquet"
    df_transformed.write.mode("overwrite").format("parquet").saveAsTable(permanent_table_name)

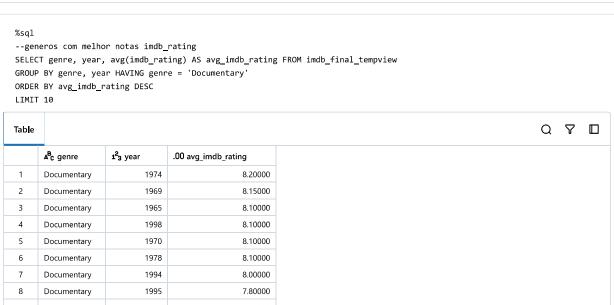
except:
    print('Tabela jé existe. Fazendo append')
    try:
        permanent_table_name = "imdb_final_dataset_csv"
        df_transformed.write.mode("append").format("parquet").saveAsTable(permanent_table_name)
    except:
        print('Erro ao fazer append.')

temp_view_name = "imdb_final_tempview"
df_transformed.createOrReplaceTempView(temp_view_name)
```

Solução dos problemas

```
# códigos para soluções
# objetivo é identificar qual genero tem a melhor nota IMDb e melhor Metascore. Obj secundário: qual ano teve a melhor média e
```





9	νοcumentary	2004	1./5000
10	Documentary	2003	7.75000
10 rows	s		

sql generos com melhor notas imdb_rating ELECT genre, avg(metascore) as avg_metascore from imdb_final_tempview roup by genre rder by avg(metascore) desc IMIT 10 ble sql generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview RULE genre, year, avg(metascore) AS metascore FROM imdb_final_tempview RULE BY genre, year AWXING genre = 'Documentary' RULE BY genre, year AMXING genre = 'Documentary' RULE BY metascore DESC IMIT 10		
sql generos com melhor notas imdb_rating ELECT genre, avg(metascore) as avg_metascore from imdb_final_tempview roup by genre rder by avg(metascore) desc IMIT 10 ble sql generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview RULE genre, year, avg(metascore) AS metascore FROM imdb_final_tempview RULE BY genre, year AWXING genre = 'Documentary' RULE BY genre, year AMXING genre = 'Documentary' RULE BY metascore DESC IMIT 10		
sql generos com melhor notas imdb_rating ELECT genre, avg(metascore) as avg_metascore from imdb_final_tempview roup by genre rder by avg(metascore) desc IMIT 10 ble sql generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview RULE genre, year, avg(metascore) AS metascore FROM imdb_final_tempview RULE BY genre, year AWXING genre = 'Documentary' RULE BY genre, year AMXING genre = 'Documentary' RULE BY metascore DESC IMIT 10		
ELECT genre, avg(metascore) as avg_metascore from imdb_final_tempview roup by genre der by avg(metascore) desc IMIT 10 ble sql -generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year AAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10	%sql	
roup by genre rder by avg(metascore) desc IMIT 10 ble sql generos com melhor notas imdb_rating ELECT genre, year avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
roup by genre rder by avg(metascore) desc IMIT 10 ble sql generos com melhor notas imdb_rating ELECT genre, year avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10	SELEC.	T genre, avg(metascore) as avg_metascore from imdb_final_tempview
sql -generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview RDUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
sql -generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
sql -generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview RQUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
sql -generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10	TIME	
sql -generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
sql -generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10	able	
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
-generos com melhor notas imdb_rating ELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10		
ROUP BY genre, year HAVING genre = 'Documentary' RDER BY metascore DESC IMIT 10	-gen	
RDER BY metascore DESC IMIT 10	SELEC.	T genre, year, avg(metascore) AS metascore FROM imdb_final_tempview
IMIT 10	GROUP	BY genre, year HAVING genre = 'Documentary'
	ORDER	BY metascore DESC
ble	LIMIT	10
ble		
ble		
	ab l e	

```
%sql
--Qual foi o genero com melhor média em cada ano

SELECT genre, year, avg_metascore FROM (
SELECT T1.*, ROW_NUMBER() OVER(PARTITION BY t1.year ORDER BY T1.avg_metascore desc) as rn
FROM(
    SELECT genre, year, avg(metascore) AS avg_metascore FROM imdb_final_tempview
    GROUP BY year, genre
    ) T1
)T2
WHERE rn = 1
ORDER BY YEAR DESC
LIMIT 100
```

Tab l e	

Respostas para Objetivos

- 1. Genero com melhor avaliação imdb_rating: Gênero Documentary
- 2. Ano com melhor nota imdb_rating do genero com melhor média: 1974 -> 8,2
- 3. Genero com melhor avaliação metascore: Gênero Documentary
- 4. Ano com melhor nota imdb_rating do genero com melhor média: 1970 -> 95,0
- 5. Tabela com o melhor genero e média para cada ano. Tabela gerado no ultimo select