databricks MVP Sprint Engenharia de Dados - Thiago Naves

(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-511eb367-a865-48fc-b3f7-4c7dd55fe8de/lib/python3.9/site-packages (1.6.14)
Requirement already satisfied: python-slugify in /local_disk0/.ephemeral_nfs/envs/pythonEnv-511eb367-a865-48fc-b3f7-4c7dd55fe8de/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-511eb367-a865-48fc-b3f7-4c7dd55fe8de/lib/python3.9/site-packages (from kaggle) (4.66.4)
Requirement already satisfied: certifi>=2023.7.22 in /local_disk0/.ephemeral_nfs/envs/pythonEnv-511eb367-a865-48fc-b3f7-4c7dd55fe8de/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-511eb367-a865-48fc-b3f7-4c7dd55fe8de/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->
```

```
def authenticate_kaggle_api(kaggle_username, kaggle_key) -> KaggleApi:
    import os

    os.environ['KAGGLE_USERNAME'] = kaggle_username
    os.environ['KAGGLE_KEY'] = kaggle_key

    from kaggle.api.kaggle_api_extended import KaggleApi

    api = KaggleApi()
    api.authenticate()
    print("API authenticated.")

    return api
```

```
# kaggle api tokens
import pandas as pd
kaggle_username = 'thalesdamiletto'
kaggle_key = '12857b21dac382a255dd1958b0928cff'
try:
    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)
Os arquivos foram baixados e extraídos para: imdb_dataset_2
```

```
display(dbutils.fs.ls('file:/databricks/driver'))
```

Table

	path	name	size	modificationTime
1	file:/databricks/driver/hadoop_accessed_config.lst	hadoop_accessed_config.lst	2755	1720647716827
2	file:/databricks/driver/azure/	azure/	4096	1720647716823
3	file:/databricks/driver/conf/	conf/	4096	1720647715883
4	file:/databricks/driver/preload_class.lst	preload_class.lst	1306936	1720647716859
5	file:/databricks/driver/imdb-dataset-2024-updated.z...	imdb-dataset-2024-updated.z...	335942	1720278268000
6	file:/databricks/driver/imdb_dataset_2/	imdb_dataset_2/	4096	1720658679412
7	file:/databricks/driver/logs/	logs/	4096	1720656819820
8	file:/databricks/driver/path/	path/	4096	1720654458483
9	file:/databricks/driver/ganglia/	ganglia/	4096	1720657802130
10	file:/databricks/driver/eventlogs/	eventlogs/	4096	1720652123845
11	file:/databricks/driver/imdb_dataset/	imdb_dataset/	4096	1720654090366
12	file:/databricks/driver/metastore_db/	metastore_db/	4096	1720652209273

12 rows

```
csv_path_file = 'file:/databricks/driver/imdb_dataset_2/*'

df_origem = spark.read.format("csv").option("header", True).load(csv_path_file)
df_origem.display()
```

Table

	Title	IMDb Rating	Year	Certificates	Genre	Director
56	Patty Hearst	6.8	1988	R	Biography	Paul Schrader
57	The Look of Silence	8.3	2014	PG-13	Documentary	Joshua Oppenheimer
58	Mesrine: Public Enemy No. 1	7.4	2008	R	Action	Jean-François Richet
59	Walker	6.6	1987	R	Biography	Alex Cox
60	The Sparks Brothers	7.7	2021	R	Documentary	Edgar Wright
61	Lady Sings the Blues	7.0	1972	R	Biography	Sidney J. Furie
62	Val	7.6	2021	R	Documentary	Ting Poo
63	Marley	7.9	2012	PG-13	Documentary	Kevin Macdonald
64	Prick Up Your Ears	7.1	1987	R	Biography	Stephen Frears
65	Lisztomania	6.1	1975	R	Biography	Ken Russell
66	Why Do Fools Fall in Love	6.4	1998	R	Biography	Gregory Nava
67	The Glorias	6.0	2020	R	Biography	Julie Taymor
68	The Story of Adele H	7.2	1975	PG	Biography	François Truffaut

69	Prefontaine	6.8	1997	PG-13	Biography	Steve James
70	Backbeat	6.6	1994	R	Biography	Iain Softley

4,573 rows

```
# atualizar código para buscar os dados da internet, ao invés da memoria do databricks link: https://www.kaggle.com/datasets/p
# df_origem = spark.read.format("csv").option("header", True).load('dbfs:///FileStore/tables/csv/mvp/*')
# df_origem.display()
```

Modelagem

```
# código de modelagem
# quebrar tabela de origem resultando em dados úteis ao nosso modelo para atender o objetivo proposto
# objetivo é identificar qual genero tem a melhor nota IMDb e melhor Metascore. Obj secundário: qual ano teve a melhor média e
# Colunas Necessáris: Title, IMDB Rating, Year, MetaScore
# Criar novo dataframe apenas com as colunas necessárias

df_cleansed = df_origem.select('Title', 'Genre', 'IMDb Rating', 'MetaScore', "Year")
df_cleansed = df_cleansed.withColumnRenamed("Title", "title").withColumnRenamed("Genre", "genre").withColumnRenamed("IMDb Rati
df_cleansed.display()
```

Table

	title	genre	imdb_rating	metascore	year
1	End of the Spear	Adventure	6.8	45.0	2005
2	Elvira Madigan	Biography	7.0	66.0	1967
3	The Kid Stays in the Picture	Documentary	7.3	75.0	2002
4	It Ain't Over	Documentary	8.2	79.0	2022
5	Mahler	Biography	7.0	66.0	1974
6	The Dark Horse	Biography	7.4	77.0	2014
7	Stephen Curry: Underrated	Documentary	7.3	69.0	2023
8	Carrington	Biography	6.8	66.0	1995
9	Burden	Biography	6.7	57.0	2018
10	Georgetown	Biography	6.2	49.0	2019
11	Miles Ahead	Biography	6.4	64.0	2015
12	The King of Kong: A Fistful of Quarters	Documentary	8.0	83.0	2007
13	The Devil's Violinist	Biography	6.0	38.0	2013
14	A Tale of Love and Darkness	Biography	6.0	55.0	2015
15	Cobb	Biography	6.4	66.0	1994

4,573 rows

```
df_cleansed.printSchema
```

```
Out[136]: <bound method DataFrame.printSchema of DataFrame[title: string, genre: string, imdb_rating: string, metascore: strin
g, year: string]>
```

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]+$"), col("imdb_rating").cast(decimal_type)).oth
    .withColumn("metascore", when(col("metascore").rlike("^[0-9]*\\.?[0-9]+$"), col("metascore").cast(decimal_type)).otherwise
    .withColumn("year", when(col("year").rlike("^[0-9]{4}$"), col("year").cast(integer_type)).otherwise(None))
    .dropna(subset=["imdb_rating", "metascore"])
)
df_transformed.display()
```

Table

	title	genre	.00 imdb_rating	.00 metascore	year
104	A United Kingdom	Biography	6.9	65.0	2016
105	The Dove	Adventure	6.3	66.0	1974
106	Lifemark	Biography	6.1	66.0	2022
107	Creation	Biography	6.6	51.0	2009
108	Permanent Midnight	Biography	6.2	57.0	1998
109	Woman Walks Ahead	Biography	6.7	51.0	2017
110	An Angel at My Table	Biography	7.4	79.0	1990
111	Won't You Be My Neighbor?	Documentary	8.3	85.0	2018
112	Mesrine: Killer Instinct	Action	7.5	71.0	2008
113	Ride Like a Girl	Biography	7.0	47.0	2019
114	The Sapphires	Biography	7.0	67.0	2012
115	Enron: The Smartest Guys in the Room	Documentary	7.6	66.0	2005
116	Swan Song	Biography	7.0	65.0	2021
117	Bird	Biography	7.1	78.0	1988
118	Queen of Katwe	Biography	7.4	73.0	2016
119	Rebel in the Rye	Biography	6.7	46.0	2017

3,573 rows

Carga de dados

```
# escrever as tabelas ou dados
try:
    permanent_table_name = "imdb_final_dataset_csv"
    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)
```

```
Tabela jé existe. Fazendo append
Erro ao fazer append.
```

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
```

```
%sql
--generos com melhor notas imdb_rating
SELECT genre, avg(imdb_rating) as avg_imdb_rating from imdb_final_tempview
group by genre
order by avg(imdb_rating) desc
--limit 10
```

Table

	genre	avg_imdb_rating
1	Documentary	7.43745
2	Thriller	7.40000
3	Crime	7.21959
4	Animation	7.20258
5	Reality-TV	7.20000
6	Biography	7.04312
7	Adventure	6.98312
8	History	6.94286
9	Action	6.89222
10	Mystery	6.85333
11	Drama	6.84591
12	Comedy	6.71273
13	Romance	6.70000
14	Musical	6.70000
15	Fantasy	6.54118

18 rows

```
%sql
--generos com melhor notas imdb_rating
SELECT genre, year, avg(imdb_rating) AS avg_imdb_rating FROM imdb_final_tempview
GROUP BY genre, year HAVING genre = 'Documentary'
ORDER BY avg_imdb_rating DESC
```

Table

	genre	year	avg_imdb_rating
29	Documentary	2010	7.38667
30	Documentary	2002	7.35000
31	Documentary	2005	7.32857
32	Documentary	2023	7.31250
33	Documentary	1982	7.30000
34	Documentary	2013	7.27500
35	Documentary	1945	7.20000
36	Documentary	1968	7.20000
37	Documentary	1985	7.10000
38	Documentary	1993	7.10000
39	Documentary	2011	7.05000
40	Documentary	2015	7.01429
41	Documentary	1991	7.00000
42	Documentary	1997	6.60000
43	Documentary	1973	6.60000
44	Documentary	1971	5.80000

44 rows

```
%sql
--generos com melhor notas imdb_rating
SELECT genre, avg(metascore) as avg_metascore from imdb_final_tempview
group by genre
order by avg(metascore) desc
--limit 10
```

Table

	genre	avg_metascore
1	Documentary	72.16733
2	Animation	70.17194
3	Crime	69.43711
4	Mystery	69.40000
5	Thriller	69.00000
6	Drama	66.94909
7	Sci-Fi	66.01970
8	Romance	66.00000
9	Reality-TV	66.00000
10	Musical	66.00000
11	Adventure	65.78397
12	Comedy	65.67818
13	Biography	64.79128
14	Fantasy	64.31471
15	Horror	64.12705

18 rows

```
%sql
--generos com melhor notas imdb_rating
SELECT genre, year, avg(metascore) AS metascore FROM imdb_final_tempview
GROUP BY genre, year HAVING genre = 'Documentary'
ORDER BY metascore DESC
```

Table

	genre	year	metascore
29	Documentary	1974	68.00000
30	Documentary	1956	66.00000
31	Documentary	1997	66.00000
32	Documentary	1965	66.00000
33	Documentary	1985	66.00000
34	Documentary	1998	66.00000
35	Documentary	1982	66.00000
36	Documentary	1971	66.00000
37	Documentary	1991	66.00000
38	Documentary	2011	66.00000
39	Documentary	1945	66.00000
40	Documentary	1968	66.00000
41	Documentary	1992	66.00000
42	Documentary	1973	66.00000
43	Documentary	1969	66.00000
44	Documentary	1995	66.00000

44 rows

```
%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 ASC
```

Table

	genre	year	avg_metascore
1	Drama	1917	66.00000
2	Sci-Fi	1918	66.00000
3	Sci-Fi	1924	66.00000
4	Biography	1927	66.00000
5	Comedy	1931	99.00000
6	Comedy	1933	66.00000
7	Comedy	1934	66.00000
8	Sci-Fi	1935	66.00000
9	Comedy	1936	96.00000
10	Drama	1937	66.00000
11	Drama	1938	66.00000
12	Adventure	1939	92.00000
13	Comedy	1940	66.00000
14	Drama	1942	100.00000
15	Adventure	1943	66.00000

94 rows

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