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-ccde3985-a79a-48d6-93ad-d5476ffefb03/lib/python3.9/site-packages (1.6.14)
Requirement already satisfied: python-slugify in /local_disk0/.ephemeral_nfs/envs/pythonEnv-ccde3985-a79a-48d6-93ad-d5476ffefb03/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/python3.9/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-d5476ffefb03/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-d5476ffefb03/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'
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/azure/	azure/	4096	1720789201980
2	file:/databricks/driver/conf/	conf/	4096	1720789201092
3	file:/databricks/driver/preload_class.lst	preload_class.lst	1306936	1720789201996
4	file:/databricks/driver/hadoop_accessed_config....	hadoop_accessed_config....	2755	1720789201992
5	file:/databricks/driver/metastore_db/	metastore_db/	4096	1720822132974
6	file:/databricks/driver/eventlogs/	eventlogs/	4096	1720821803446
7	file:/databricks/driver/imdb_dataset_2/	imdb_dataset_2/	4096	1720822827938
8	file:/databricks/driver/ganglia/	ganglia/	4096	1720822503578
9	file:/databricks/driver/logs/	logs/	4096	1720822128506

9 rows

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

df_origem = spark.read.format("csv").option("header", True).load(csv_path_file)
display(df_origem.limit(5))
```

Table

	Title	IMDb Rating	Year	Certificates	Genre	Director	Star Cast
1	End of the Spear	6.8	2005	PG-13	Adventure	Jim Hanon	Louie LeonardoC
2	Elvira Madigan	7.0	1967	PG	Biography	Bo Widerberg	Pia DegermarkTh
3	The Kid Stays in the Pictu...	7.3	2002	R	Documentary	Nanette Burstein	Robert EvansEdd
4	It Ain't Over	8.2	2022	PG	Documentary	Sean Mullin	Andy AndresRog

5 rows

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
display(df_cleansed.limit(5))
```

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 Pictu...	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

5 rows

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"])
)
display(df_transformed.limit(5))
```

Table

	title	genre	.00 imdb_rating	.00 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 Pictu...	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

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
```

```
%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

10 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
LIMIT 10
```

Table

	genre	year	avg_imdb_rating
1	Documentary	1974	8.20000
2	Documentary	1969	8.15000
3	Documentary	1965	8.10000
4	Documentary	1998	8.10000
5	Documentary	1970	8.10000
6	Documentary	1978	8.10000
7	Documentary	1994	8.00000
8	Documentary	1995	7.80000

9	Documentary	2004	7.75000
10	Documentary	2003	7.75000

10 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

```
%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
LIMIT 10
```

Table

```
%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
```

Table

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