



MVP ENGENHARIA DE DADOS

VICTOR VIDAL – PUC-RJ

ETAPA I – OBJETIVO

OBJETIVO

A partir de bases de dados extraídas do site IMDb (Internet Movie Database), analisar:

- Quais são os 10 filmes com as maiores notas de avaliação dos usuários?
- Quais são os 10 filmes mais populares (com maior números de votos)?

ETAPA 2 – COLETA DOS DADOS

Busca pelos dados

Site: datasets.imdbws.com

Coleta

Os dados foram baixados do site do IMDb para máquina local.

IMDb data files available for download

Documentation for these data files can be found on <http://www.imdb.com/interfaces/>

[name.basics.tsv.gz](#)

[title.akas.tsv.gz](#)

[title.basics.tsv.gz](#)

[title.crew.tsv.gz](#)

[title.episode.tsv.gz](#)

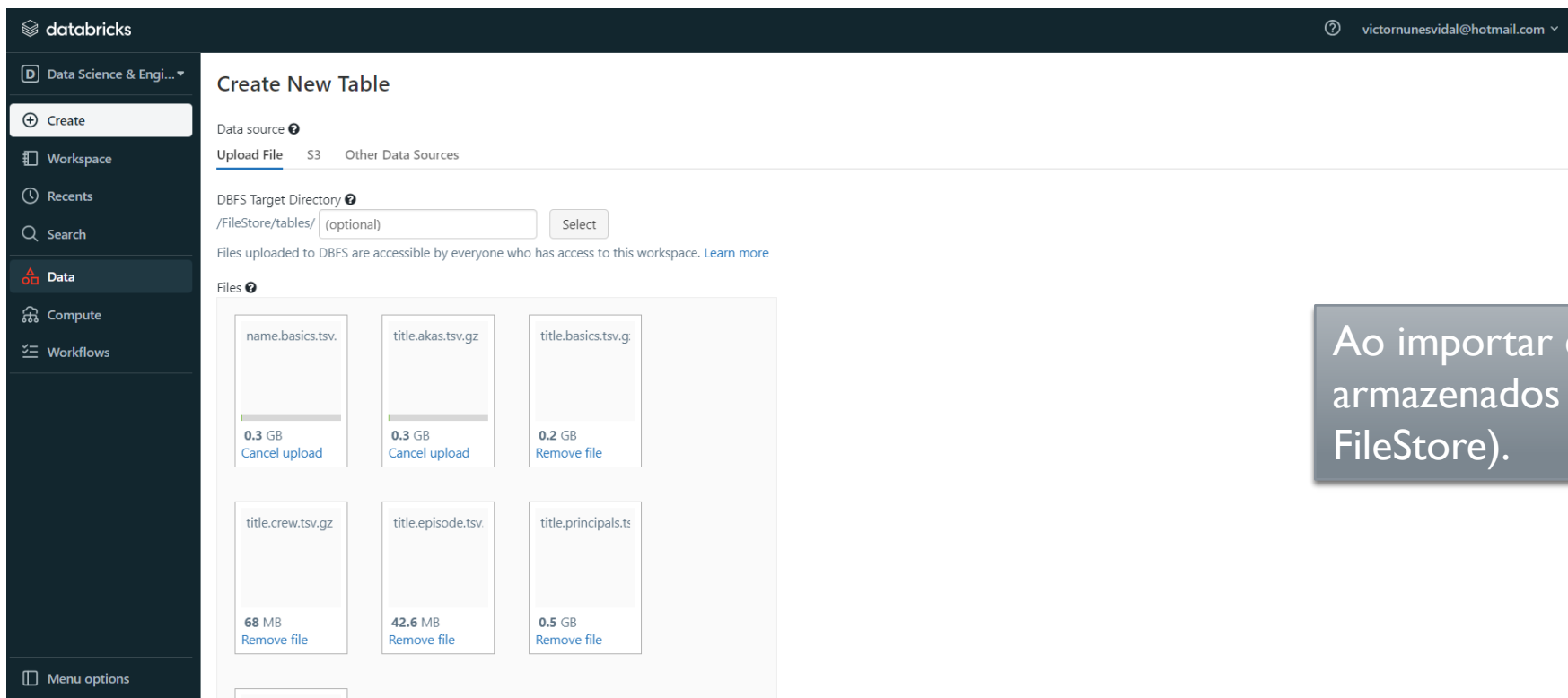
[title.principals.tsv.gz](#)

[title.ratings.tsv.gz](#)

ETAPA 3 – CRIAÇÃO DE CLUSTER NO DATABRICKS

The screenshot displays the Databricks web interface. On the left is a dark sidebar with navigation options: 'Data Science & Engi...', 'Create', 'Workspace', 'Recents', 'Search', 'Data', 'Compute' (highlighted), 'Workflows', and 'Menu options'. The top header shows the 'databricks' logo and the user 'victormunesvidal@hotmail.com'. The main content area is titled 'Compute >' and features a cluster named 'VictorCluster' with a green status icon. To the right of the cluster name are buttons for 'More', 'Terminate', and 'Edit'. Below the cluster name is a horizontal tab bar with 'Configuration' (selected), 'Notebooks (0)', 'Libraries', 'Event log', 'Spark UI', 'Driver logs', 'Metrics', 'Apps', and 'Spark compute UI - Master'. The 'Configuration' tab shows the following settings: 'Databricks Runtime Version' set to '12.2 LTS (includes Apache Spark 3.3.2, Scala 2.12)', 'Driver type' set to 'Community Optimized' with '15.3 GB Memory, 2 Cores, 1 DBU', and an 'Instance' section with a warning: 'Free 15 GB Memory: As a Community Edition user, your compute will automatically terminate after an idle period of one or two hours. For [more configuration options](#), please [upgrade your Databricks subscription](#).' Below this is a tab bar for 'Instances', 'Spark', and 'JDBC/ODBC'. The 'Instances' tab shows the 'Availability zone' set to 'us-west-2c'.

ETAPA 4 – UPLOAD DE ARQUIVOS NO DATABRICKS

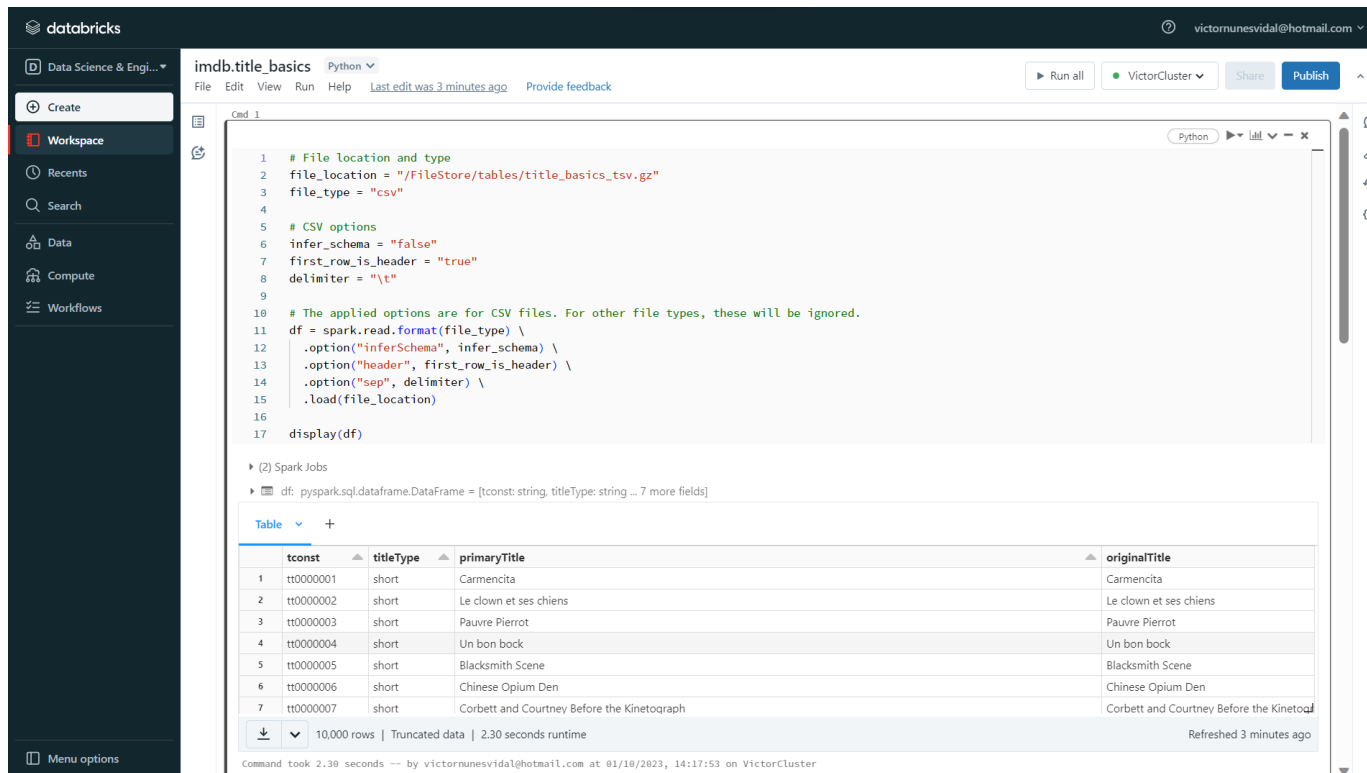


The screenshot shows the Databricks web interface. The top navigation bar includes the Databricks logo, a user profile icon, and the email address 'victornunesvidal@hotmail.com'. The left sidebar contains navigation links for 'Data Science & Engi...', 'Create', 'Workspace', 'Recents', 'Search', 'Data', 'Compute', 'Workflows', and 'Menu options'. The main content area is titled 'Create New Table' and features a 'Data source' dropdown set to 'Upload File'. Below this, the 'DBFS Target Directory' is set to '/FileStore/tables/' with a 'Select' button. A note states: 'Files uploaded to DBFS are accessible by everyone who has access to this workspace. [Learn more](#)'. The 'Files' section displays six upload progress cards:

File Name	Size	Upload Status
name.basics.tsv.	0.3 GB	Cancel upload
title.akas.tsv.gz	0.3 GB	Cancel upload
title.basics.tsv.g.	0.2 GB	Remove file
title.crew.tsv.gz	68 MB	Remove file
title.episode.tsv.	42.6 MB	Remove file
title.principals.ts	0.5 GB	Remove file

Ao importar esses arquivos, eles ficam armazenados no DBFS (Databricks FileStore).

ETAPA 5 – NOTEBOOK PARA CRIAÇÃO DE TABELA NO DATABRICKS



The screenshot shows a Databricks workspace with a notebook named 'imdb.title_basics'. The notebook contains Python code that reads a TSV file from the FileStore and displays the resulting DataFrame. Below the code, a table preview is shown with 7 rows and 4 columns: tconst, titleType, primaryTitle, and originalTitle. The table contains data for various movies, including 'Carmencita', 'Le clown et ses chiens', 'Pauvre Pierrot', 'Un bon bock', 'Blacksmith Scene', 'Chinese Opium Den', and 'Corbett and Courtney Before the Kinetograph'.

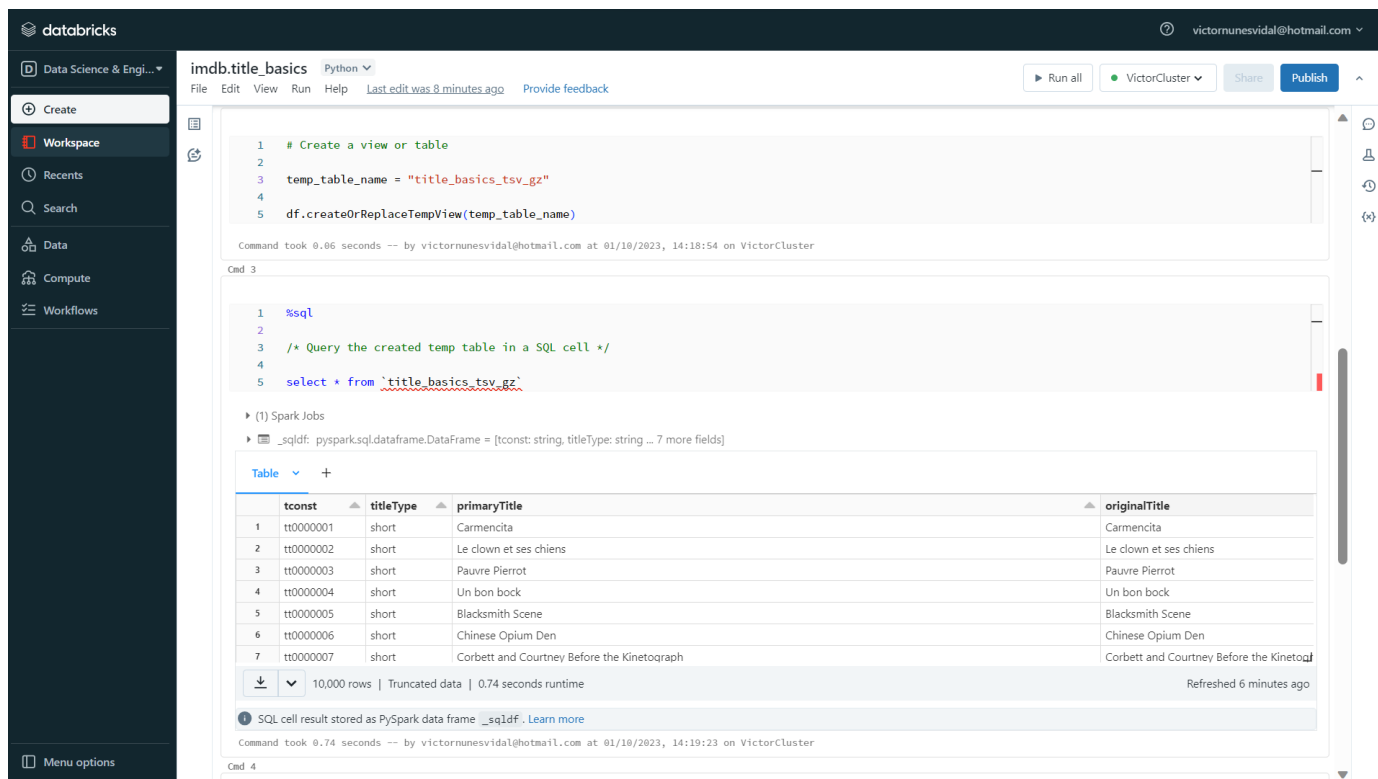
```
1 # File location and type
2 file_location = "/FileStore/tables/title_basics_tsv.gz"
3 file_type = "csv"
4
5 # CSV options
6 infer_schema = "false"
7 first_row_is_header = "true"
8 delimiter = "\t"
9
10 # The applied options are for CSV files. For other file types, these will be ignored.
11 df = spark.read.format(file_type) \
12     .option("inferSchema", infer_schema) \
13     .option("header", first_row_is_header) \
14     .option("sep", delimiter) \
15     .load(file_location)
16
17 display(df)
```

	tconst	titleType	primaryTitle	originalTitle
1	tt0000001	short	Carmencita	Carmencita
2	tt0000002	short	Le clown et ses chiens	Le clown et ses chiens
3	tt0000003	short	Pauvre Pierrot	Pauvre Pierrot
4	tt0000004	short	Un bon bock	Un bon bock
5	tt0000005	short	Blacksmith Scene	Blacksmith Scene
6	tt0000006	short	Chinese Opium Den	Chinese Opium Den
7	tt0000007	short	Corbett and Courtney Before the Kinetograph	Corbett and Courtney Before the Kinetograph

10,000 rows | Truncated data | 2.30 seconds runtime
Command took 2.30 seconds -- by victornunesvidal@hotmail.com at 01/10/2023, 14:17:53 on VictorCluster

Primeira etapa faz a ingestão do arquivo. Os arquivos utilizados do site do IMDb estão zipados em um arquivo .gz e dentro deles, há um arquivo .tsv. Para que o databricks reconheça esse formato, importei o arquivo como se fosse csv, mas o separador, ao invés de ser vírgula, foi “\t”, que é o tipo de separador do arquivo tsv.

ETAPA 6 – NOTEBOOK PARA CRIAÇÃO DE TABELA NO DATABRICKS



The screenshot shows a Databricks notebook interface. The notebook is titled "imdb.title_basics" and is running Python code. The code creates a temporary table named "title_basics_tsv_gz" and then queries it using SQL. The output of the SQL query is displayed as a table with 7 rows and 4 columns: tconst, titleType, primaryTitle, and originalTitle. The table contains data for various movies, including "Carmencita", "Le clown et ses chiens", "Pauvre Pierrot", "Un bon bock", "Blacksmith Scene", "Chinese Opium Den", and "Corbett and Courtney Before the Kinetograph".

```
1 # Create a view or table
2
3 temp_table_name = "title_basics_tsv_gz"
4
5 df.createOrReplaceTempView(temp_table_name)
```

Command took 0.06 seconds -- by victornunesvidal@hotmail.com at 01/18/2023, 14:18:54 on VictorCluster

```
1 %sql
2
3 /* Query the created temp table in a SQL cell */
4
5 select * from `title_basics_tsv_gz`
```

(1) Spark Jobs

_sqlidf: pyspark.sql.dataframe.DataFrame = [tconst: string, titleType: string ... 7 more fields]

	tconst	titleType	primaryTitle	originalTitle
1	tt0000001	short	Carmencita	Carmencita
2	tt0000002	short	Le clown et ses chiens	Le clown et ses chiens
3	tt0000003	short	Pauvre Pierrot	Pauvre Pierrot
4	tt0000004	short	Un bon bock	Un bon bock
5	tt0000005	short	Blacksmith Scene	Blacksmith Scene
6	tt0000006	short	Chinese Opium Den	Chinese Opium Den
7	tt0000007	short	Corbett and Courtney Before the Kinetograph	Corbett and Courtney Before the Kinetograph

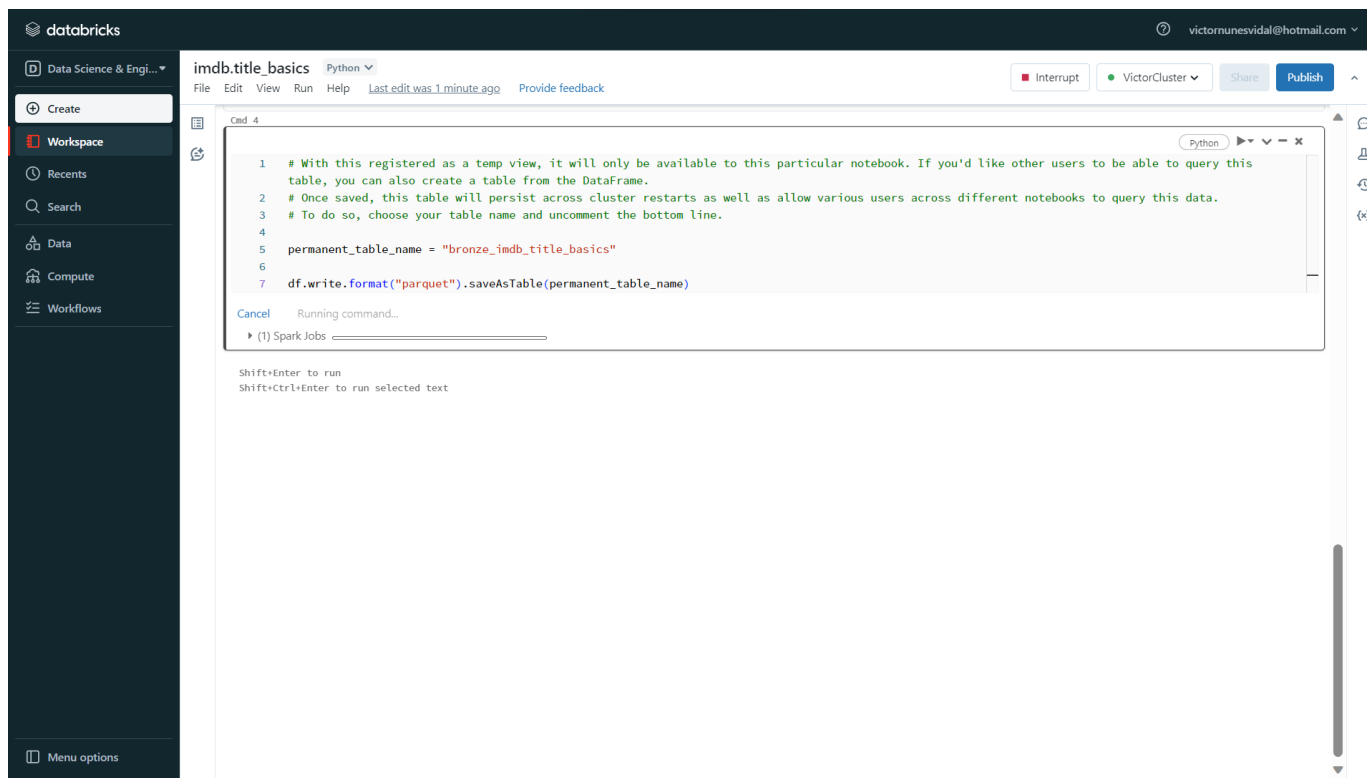
10,000 rows | Truncated data | 0.74 seconds runtime | Refreshed 6 minutes ago

SQL cell result stored as PySpark data frame _sqlidf. Learn more

Command took 0.74 seconds -- by victornunesvidal@hotmail.com at 01/18/2023, 14:19:23 on VictorCluster

Depois é criada uma tabela temporária para armazenar os dados. O SELECT mostra os dados que estão armazenados nela.

ETAPA 7 – NOTEBOOK PARA CRIAÇÃO DE TABELA NO DATABRICKS



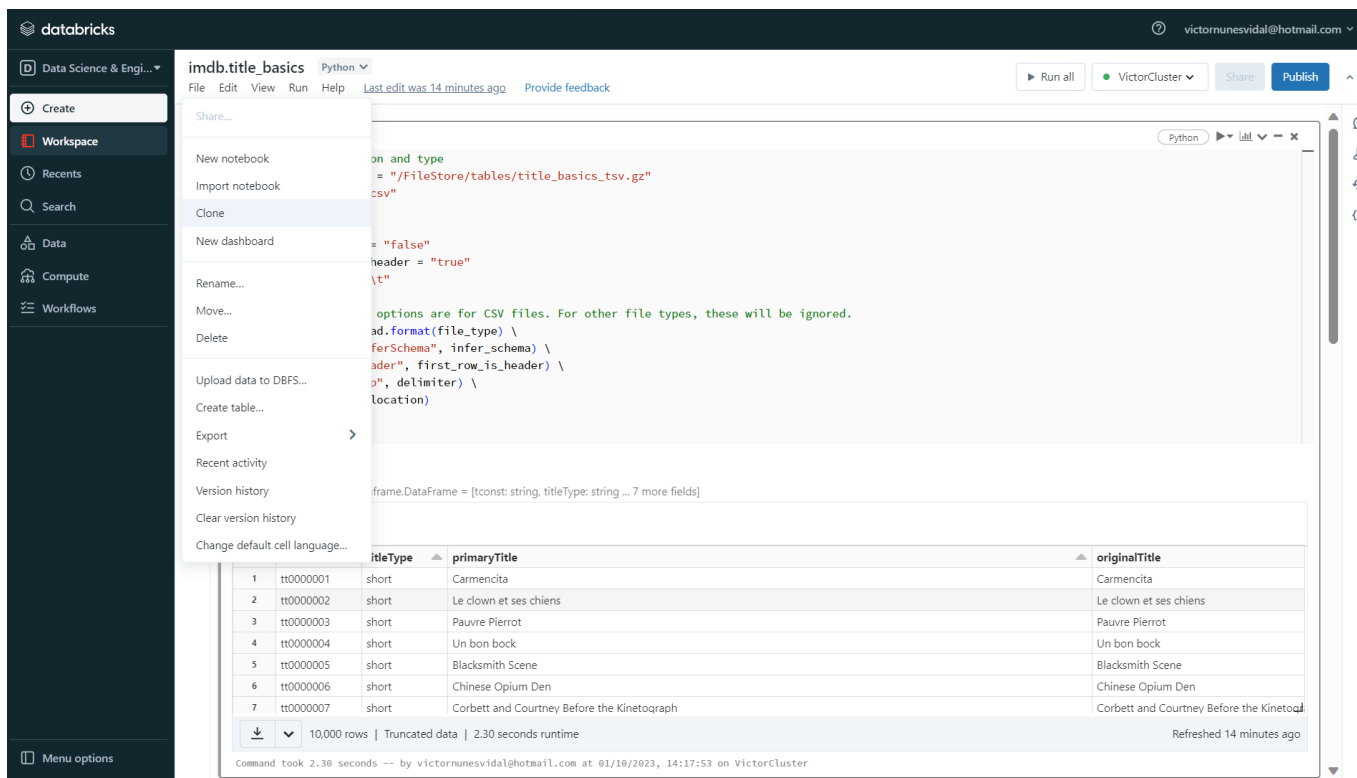
The screenshot shows the Databricks interface with a notebook titled 'imdb.title_basics'. The code in the notebook is as follows:

```
1 # With this registered as a temp view, it will only be available to this particular notebook. If you'd like other users to be able to query this
2 # table, you can also create a table from the DataFrame.
3 # Once saved, this table will persist across cluster restarts as well as allow various users across different notebooks to query this data.
4 # To do so, choose your table name and uncomment the bottom line.
5 permanent_table_name = "bronze_imdb_title_basics"
6
7 df.write.format("parquet").saveAsTable(permanent_table_name)
```

Below the code editor, there is a status bar indicating 'Running command...' and a 'Cancel' button. At the bottom, there are instructions: 'Shift+Enter to run' and 'Shift+Ctrl+Enter to run selected text'.

Por último, é criada uma tabela permanente com os dados da tabela temporária.

ETAPA 8 – NOTEBOOK SALVO E DUPLICADO



The screenshot shows the Databricks workspace interface. A notebook titled 'imdb.title_basics' is open, showing Python code that reads a TSV file from the FileStore. The 'File' menu is open, and the 'Clone' option is highlighted. Below the code, a table displays the data loaded from the file.

	titleType	primaryTitle	originalTitle
1	tt0000001	short	Carmencita
2	tt0000002	short	Le clown et ses chiens
3	tt0000003	short	Pauvre Pierrot
4	tt0000004	short	Un bon bock
5	tt0000005	short	Blacksmith Scene
6	tt0000006	short	Chinese Opium Den
7	tt0000007	short	Corbett and Courtney Before the Kinetograph

Command took 2.38 seconds -- by victornunesvidal@hotmail.com at 01/10/2023, 14:17:53 on VictorCluster

No Databricks, em File > Clone, é duplicado o notebook e feito o mesmo processo para o outro arquivo que será utilizado nesse trabalho, que é o arquivo que contém as notas e número de votos de cada título do IMDb.

ETAPA 9 – DATA QUALITY

default.bronze_imdb_title_basics |

 Refresh

VictorCluster | ▾

Schema:

	col_name ▲	data_type ▲	comment ▲	
1	tconst	string	null	
2	titleType	string	null	
3	primaryTitle	string	null	
4	originalTitle	string	null	
5	isAdult	string	null	
6	startYear	string	null	
7	endYear	string	null	

default.bronze_imdb_title_ratings |

 Refresh

VictorCluster | ▾

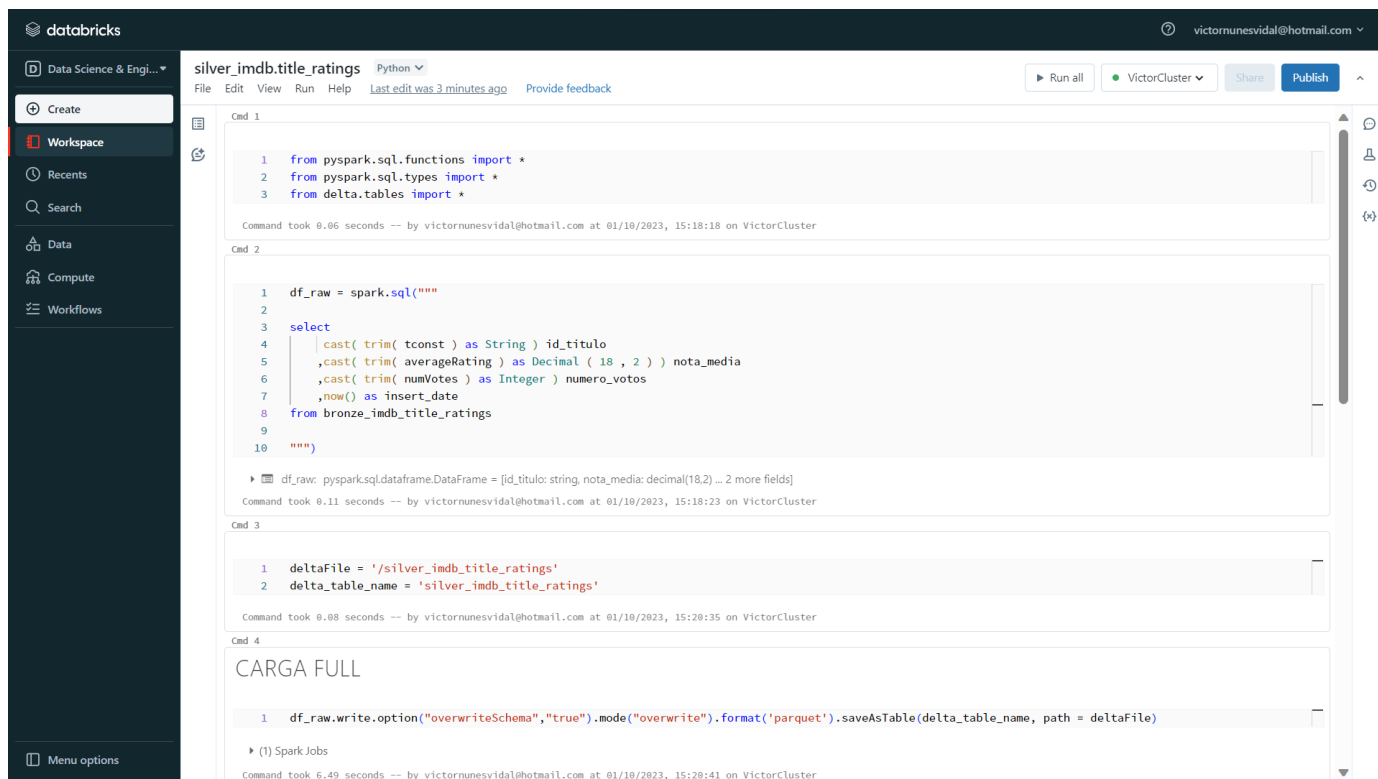
Schema:

	col_name ▲	data_type ▲	comment ▲	
1	tconst	string	null	
2	averageRating	string	null	
3	numVotes	string	null	

Ao checar os schemas das duas tabelas importadas, vemos que as colunas estão em formato string. Isso já era esperado, já que não fizemos nenhum tratamento de data type.

Dessa forma, vamos criar dois novos notebooks para essas tabelas bronze, que serão as camadas silver dessas tabelas, onde faremos os tratamentos de data type.

ETAPA 10 – CAMADA SILVER



The screenshot shows the Databricks workspace interface with the following commands:

```
Cmd 1
1 from pyspark.sql.functions import *
2 from pyspark.sql.types import *
3 from delta.tables import *

Command took 0.06 seconds -- by victornunesvidal@hotmail.com at 01/10/2023, 15:18:18 on VictorCluster

Cmd 2
1 df_raw = spark.sql("""
2
3 select
4   | cast( trim( tconst ) as String ) id_titulo
5   , cast( trim( averageRating ) as Decimal ( 18 , 2 ) ) nota_media
6   , cast( trim( numVotes ) as Integer ) numero_votos
7   , now() as insert_date
8 from bronze_imdb_title_ratings
9
10 """)

df_raw: pyspark.sql.dataframe.DataFrame = [id_titulo: string, nota_media: decimal(18,2) ... 2 more fields]
Command took 0.11 seconds -- by victornunesvidal@hotmail.com at 01/10/2023, 15:18:23 on VictorCluster

Cmd 3
1 deltaFile = '/silver_imdb_title_ratings'
2 delta_table_name = 'silver_imdb_title_ratings'

Command took 0.08 seconds -- by victornunesvidal@hotmail.com at 01/10/2023, 15:20:35 on VictorCluster

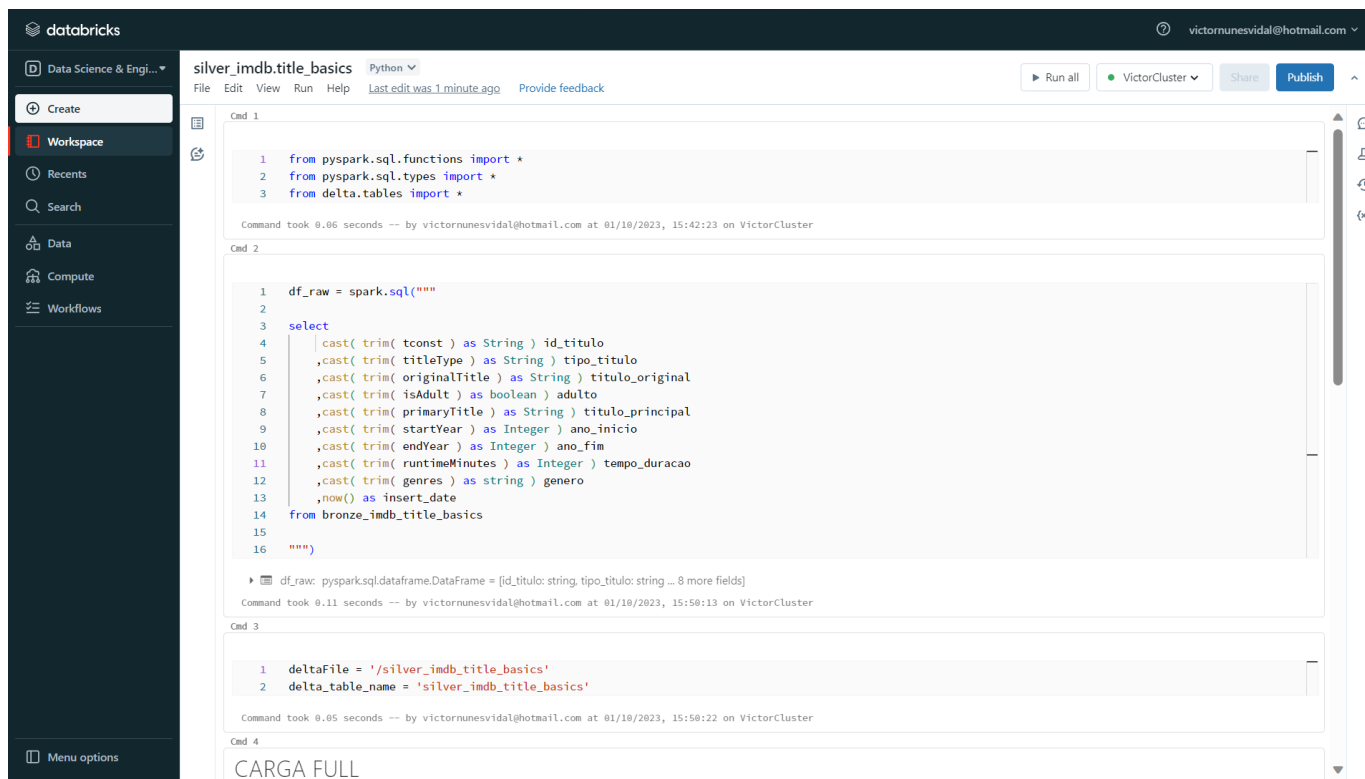
Cmd 4
CARGA FULL

1 df_raw.write.option("overwriteSchema", "true").mode("overwrite").format('parquet').saveAsTable(delta_table_name, path = deltaFile)

(1) Spark Jobs
Command took 6.49 seconds -- by victornunesvidal@hotmail.com at 01/10/2023, 15:20:41 on VictorCluster
```

- Importadas bibliotecas;
- Feito CAST para cada um dos campos indicando o formato correto deles, bem como renomeando-os;
- Utilizada função TRIM para corrigir quaisquer inconsistências comuns em bancos de dados (Exemplo: espaços em branco no final ou no começo do campo);
- Feita carga em tabela silver.
- Os datatypes utilizados aqui foram feitos com base na documentação do site do IMDb: IMDb Non-Commercial Datasets

ETAPA II – CAMADA SILVER



silver_imdb.title_basics Python

File Edit View Run Help Last edit was 1 minute ago Provide feedback

Run all VictorCluster Share Publish

Cmd 1

```
1 from pyspark.sql.functions import *
2 from pyspark.sql.types import *
3 from delta.tables import *
```

Command took 0.06 seconds -- by victornunesvidal@hotmail.com at 01/10/2023, 15:42:23 on VictorCluster

Cmd 2

```
1 df_raw = spark.sql("""
2
3 select
4     | cast( trim( tconst ) as String ) id_titulo
5     , cast( trim( titleType ) as String ) tipo_titulo
6     , cast( trim( originalTitle ) as String ) titulo_original
7     , cast( trim( isAdult ) as boolean ) adulto
8     , cast( trim( primaryTitle ) as String ) titulo_principal
9     , cast( trim( startYear ) as Integer ) ano_inicio
10    , cast( trim( endYear ) as Integer ) ano_fim
11    , cast( trim( runtimeMinutes ) as Integer ) tempo_duracao
12    , cast( trim( genres ) as string ) genero
13    , now() as insert_date
14 from bronze_imdb_title_basics
15
16 """)
```

df_raw: pyspark.sql.dataframe.DataFrame = [id_titulo: string, tipo_titulo: string ... 8 more fields]

Command took 0.11 seconds -- by victornunesvidal@hotmail.com at 01/10/2023, 15:50:13 on VictorCluster

Cmd 3

```
1 deltaFile = '/silver_imdb_title_basics'
2 delta_table_name = 'silver_imdb_title_basics'
```

Command took 0.05 seconds -- by victornunesvidal@hotmail.com at 01/10/2023, 15:50:22 on VictorCluster

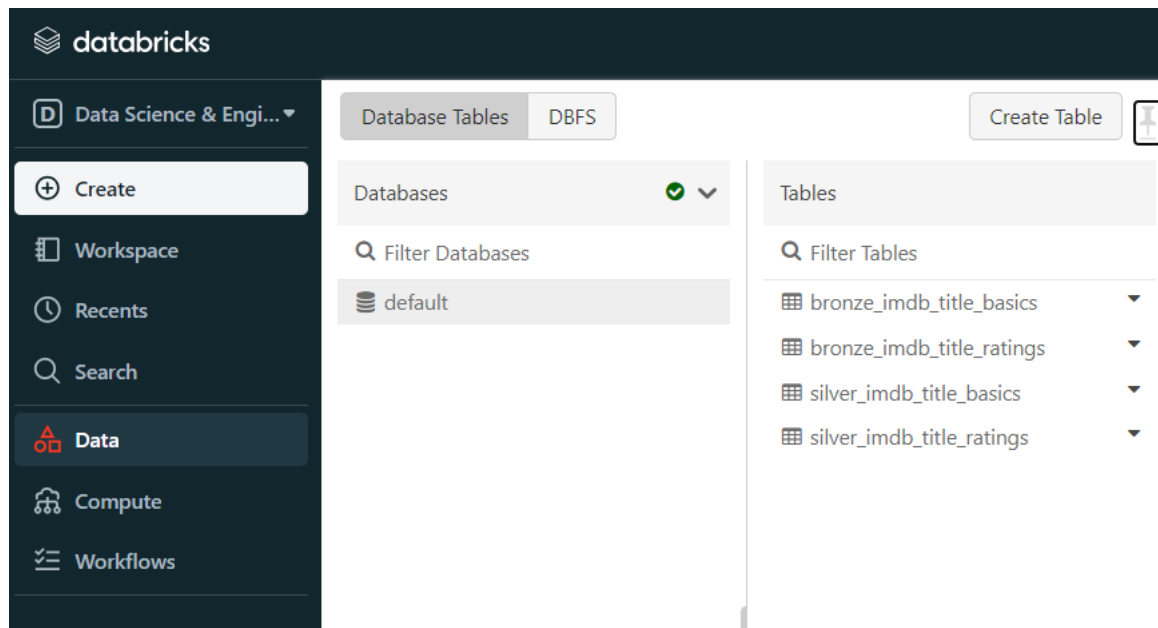
Cmd 4

CARGA FULL

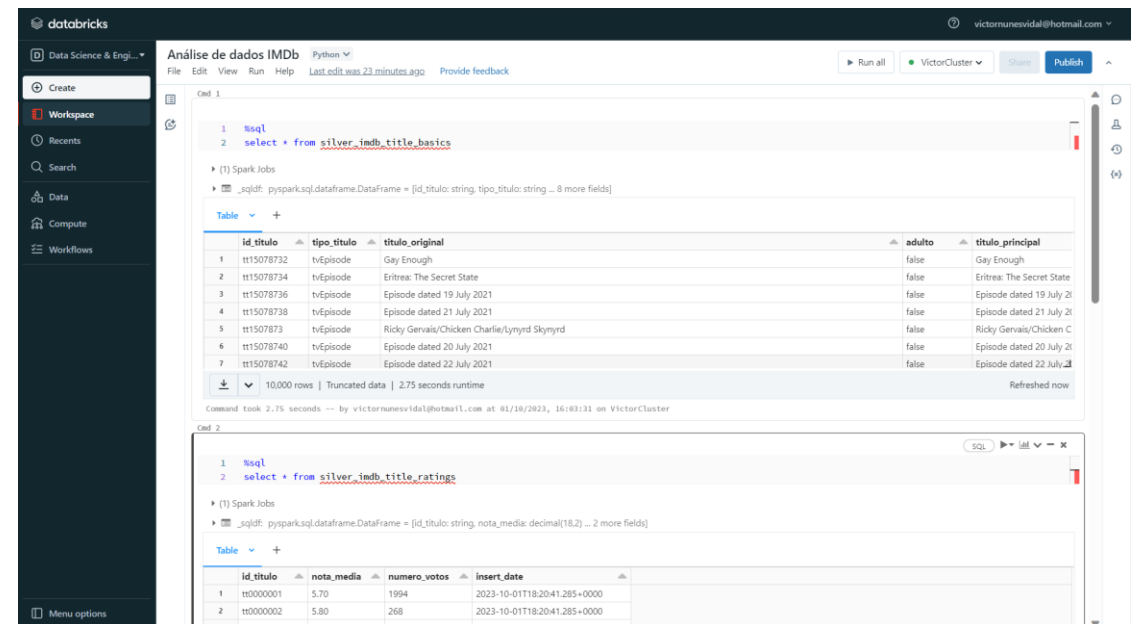
- Mesmo processo feito para tabela `imdb_title_basics`

ETAPA II – CONFERÊNCIAS

Na interface do Databricks, conferimos que as tabelas já estão disponíveis para consulta:



Fazendo SELECT nas tabelas da camada silver, também já visualizamos os dados:



ETAPA 12 – ANÁLISE DOS DADOS

The screenshot shows the Databricks interface with a workspace titled "Análise de dados IMDb". A SQL query is executed, and the results are displayed in a table. The query filters for movies with a rating of 10.00 and orders them by the number of votes in descending order. The results table shows 7 rows of data, including movie titles like "Minecraft: Apocalypse Zumbi 2" and "Girls Loving Girls".

```
1 %sql
2 select * from silver_imdb_title_ratings r
3 inner join silver_imdb_title_basics b ON b.id_titulo = r.id_titulo AND b.tipo_titulo = 'movie'
4 order by nota_media desc
```

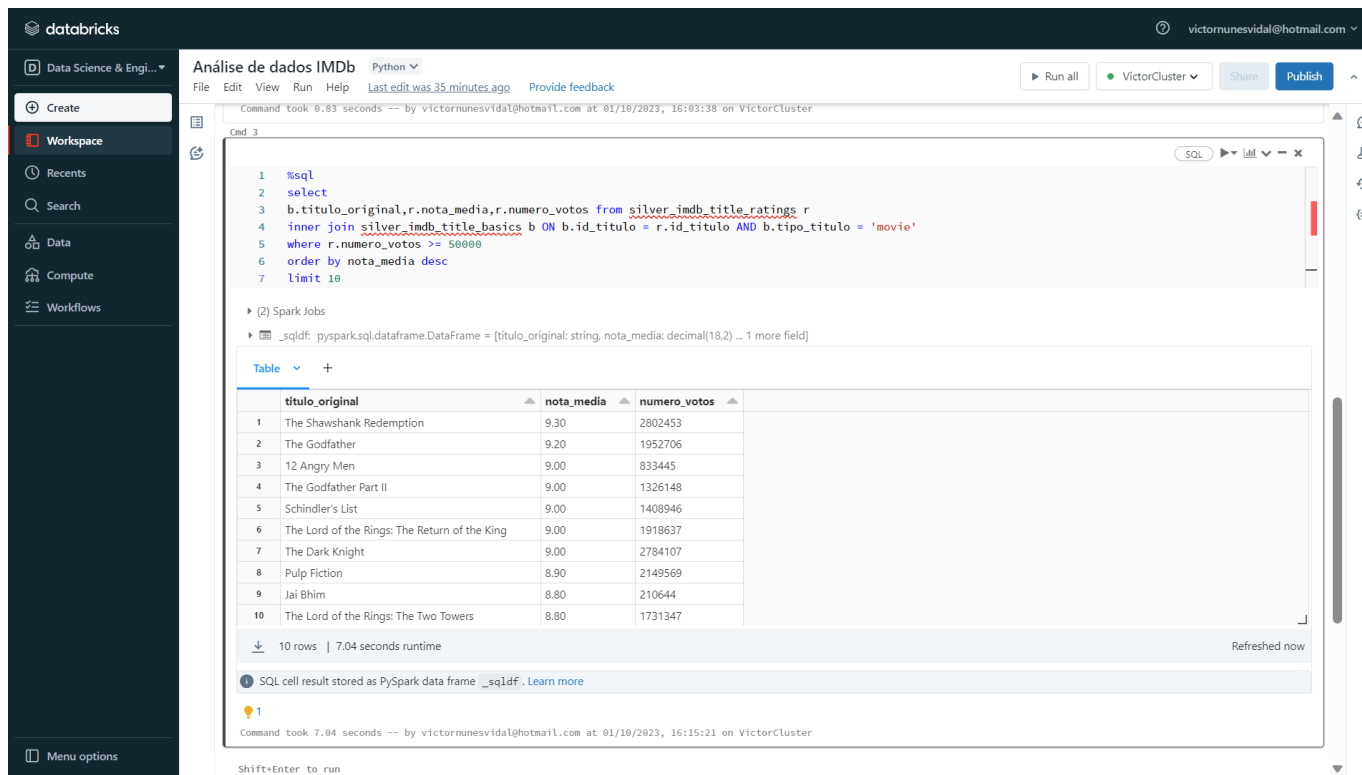
	id_titulo	nota_media	numero_votos	insert_date	id_titulo	tipo_titulo	titulo_original
1	tt4188272	10.00	7	2023-10-01T18:20:41.285+0000	tt4188272	movie	Minecraft: Apocalypse Zumbi 2
2	tt0160316	10.00	14	2023-10-01T18:20:41.285+0000	tt0160316	movie	Girls Loving Girls
3	tt15108182	10.00	6	2023-10-01T18:20:41.285+0000	tt15108182	movie	Ashes to Ashes
4	tt10449358	10.00	7	2023-10-01T18:20:41.285+0000	tt10449358	movie	Kaputol
5	tt4428002	10.00	10	2023-10-01T18:20:41.285+0000	tt4428002	movie	I Chose Life: Stories of Suicide and Survival
6	tt10463270	10.00	5	2023-10-01T18:20:41.285+0000	tt10463270	movie	Shouting Silence
7	tt15287644	10.00	20	2023-10-01T18:20:41.285+0000	tt15287644	movie	Saint-Petersburg, Gatchina, Tsarskoe selo

10,000 rows | Truncated data | 15.18 seconds runtime | Refreshed 9 minutes ago

Ao fazer a primeira query para trazer os filmes com maior nota, percebi que apareceram filmes bem desconhecidos e com pouquíssimas avaliações.

Dessa forma, vamos estabelecer um critério para as maiores notas: somente filmes com mais de 50 mil avaliações.

ETAPA 13 – ANÁLISE DOS DADOS



The screenshot shows the Databricks interface with a workspace titled "Análise de dados IMDb". The SQL query is as follows:

```
1 %sql
2 select
3 b.titulo_original,r.nota_media,r.numero_votos from silver_imdb.title_ratings r
4 inner join silver_imdb.title_basics b ON b.id_titulo = r.id_titulo AND b.tipo_titulo = 'movie'
5 where r.numero_votos >= 50000
6 order by nota_media desc
7 limit 10
```

The results table displays the top 10 movies by average rating and number of votes:

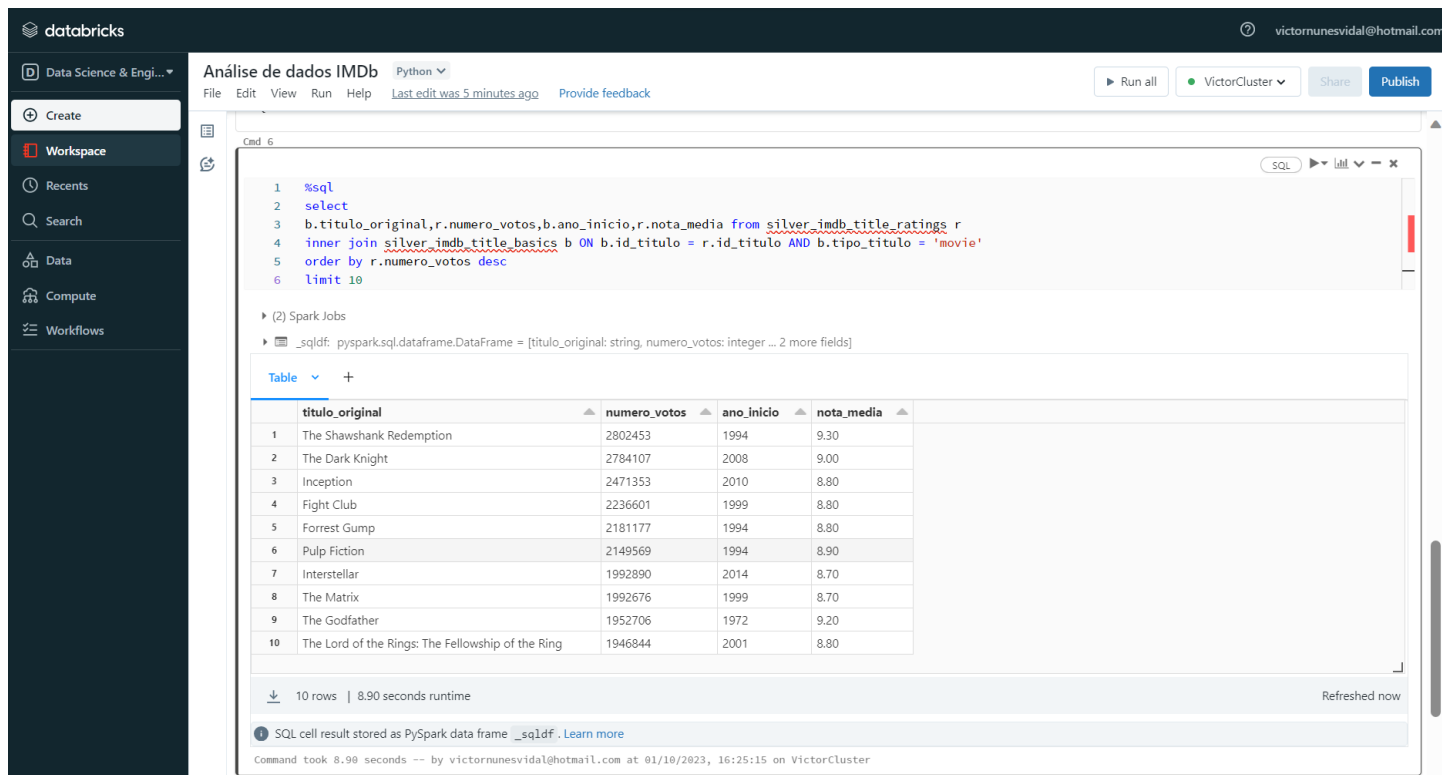
	titulo_original	nota_media	numero_votos
1	The Shawshank Redemption	9.30	2802453
2	The Godfather	9.20	1952706
3	12 Angry Men	9.00	833445
4	The Godfather Part II	9.00	1326148
5	Schindler's List	9.00	1408946
6	The Lord of the Rings: The Return of the King	9.00	1918637
7	The Dark Knight	9.00	2784107
8	Pulp Fiction	8.90	2149569
9	Jai Bhim	8.80	210644
10	The Lord of the Rings: The Two Towers	8.80	1731347

Below the table, it indicates "10 rows | 7.04 seconds runtime" and "Refreshed now". A message states: "SQL cell result stored as PySpark data frame _sqldf. Learn more". The command execution time is noted as "Command took 7.04 seconds -- by victornunesvidal@hotmail.com at 01/10/2023, 16:15:21 on VictorCluster".

Agora sim, ao colocar um filtro, trazemos os 10 filmes com as maiores notas.

O primeiro colocado é o filme “The Shawshank Redemption” (no Brasil, “Um Sonho de Liberdade”), lançado em 1994. O filme foi avaliado por 2,8 milhões de usuários no IMDb.

ETAPA 14 – ANÁLISE DOS DADOS



The screenshot shows the Databricks interface with a workspace titled "Análise de dados IMDb". The SQL query in the editor is as follows:

```
1 %sql
2 select
3   b.titulo_original, r.numero_votos, b.ano_inicio, r.nota_media from silver_imdb.title_ratings r
4   inner join silver_imdb.title_basics b ON b.id_titulo = r.id_titulo AND b.tipo_titulo = 'movie'
5   order by r.numero_votos desc
6   limit 10
```

Below the query, the results are displayed in a table format:

	titulo_original	numero_votos	ano_inicio	nota_media
1	The Shawshank Redemption	2802453	1994	9.30
2	The Dark Knight	2784107	2008	9.00
3	Inception	2471353	2010	8.80
4	Fight Club	2236601	1999	8.80
5	Forrest Gump	2181177	1994	8.80
6	Pulp Fiction	2149569	1994	8.90
7	Interstellar	1992890	2014	8.70
8	The Matrix	1992676	1999	8.70
9	The Godfather	1952706	1972	9.20
10	The Lord of the Rings: The Fellowship of the Ring	1946844	2001	8.80

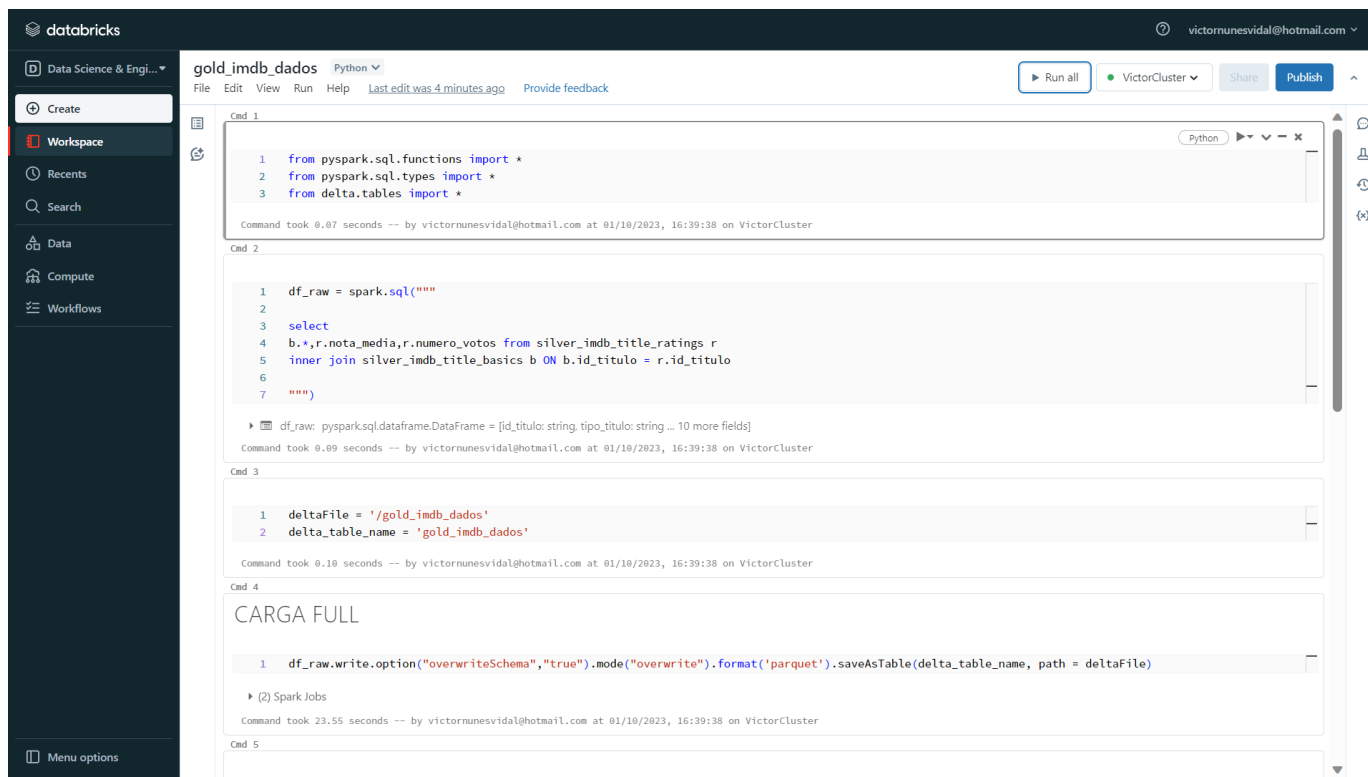
The interface also shows a sidebar with navigation options like "Create", "Workspace", "Recents", "Search", "Data", "Compute", and "Workflows". At the bottom, it indicates that the command took 8.90 seconds to run on the VictorCluster.

Para responder à 2ª pergunta proposta, sobre quais são os filmes mais populares, fazemos uma ordenação pelo campo `numero_votos`.

Quatro filmes dessa lista de mais populares também estão na lista de mais bem avaliados feita anteriormente.

O líder é o mesmo: “The Shawshank Redemption”.

ETAPA 15 – CAMADA GOLD



The screenshot shows the Databricks workspace interface. The notebook is named "gold_imdb_dados" and is running on a "VictorCluster". The code is as follows:

```
Cmd 1
1 from pyspark.sql.functions import *
2 from pyspark.sql.types import *
3 from delta.tables import *

Command took 0.07 seconds -- by victornunesvidal@hotmail.com at 01/10/2023, 16:39:38 on VictorCluster

Cmd 2
1 df_raw = spark.sql("""
2
3 select
4 b.*,r.nota_media,r.numero_votos from silver_imdb_title_ratings r
5 inner join silver_imdb_title_basics b ON b.id_titulo = r.id_titulo
6
7 """)
df_raw: pyspark.sql.dataframe.DataFrame = [id_titulo: string, tipo_titulo: string ... 10 more fields]
Command took 0.09 seconds -- by victornunesvidal@hotmail.com at 01/10/2023, 16:39:38 on VictorCluster

Cmd 3
1 deltaFile = '/gold_imdb_dados'
2 delta_table_name = 'gold_imdb_dados'

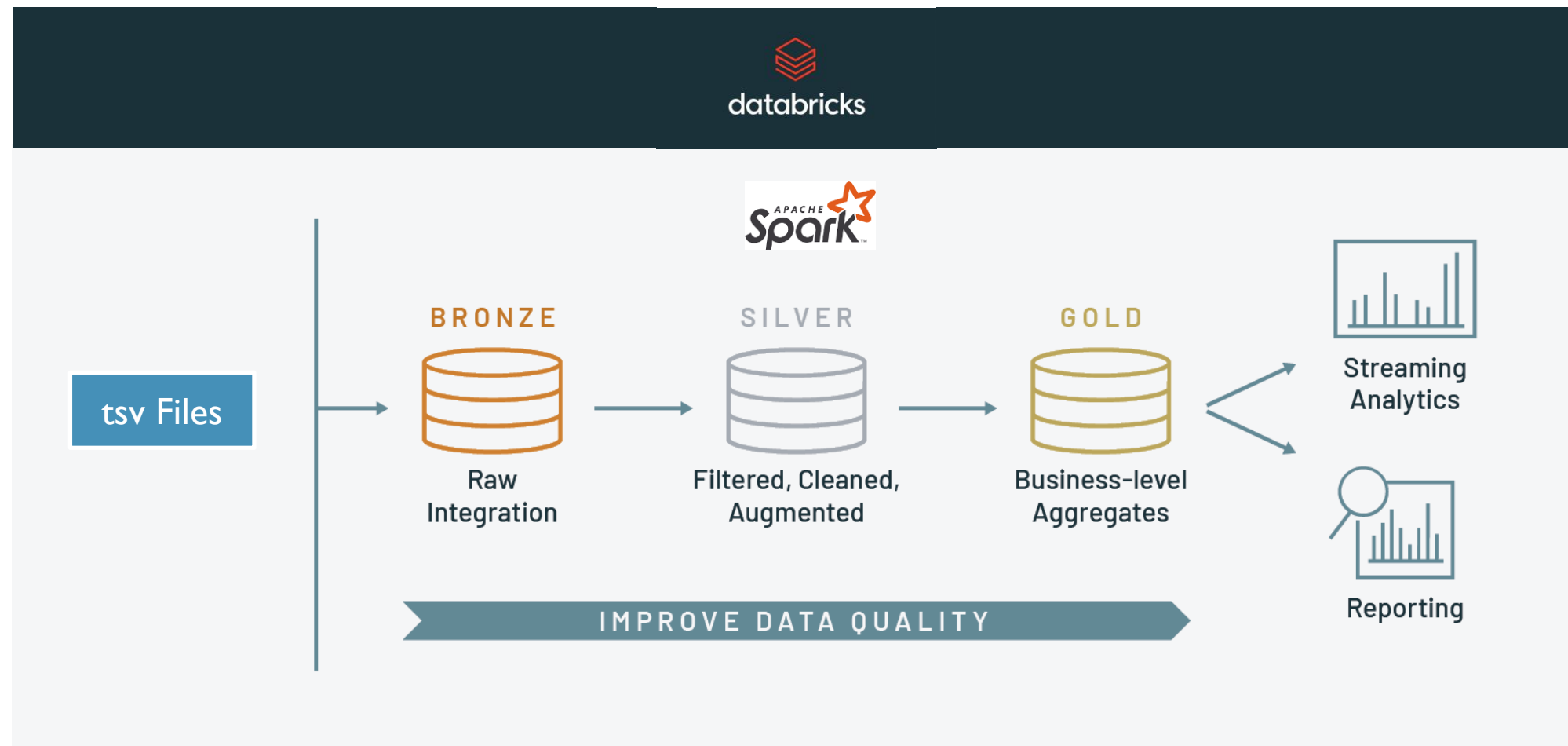
Command took 0.10 seconds -- by victornunesvidal@hotmail.com at 01/10/2023, 16:39:38 on VictorCluster

Cmd 4
CARGA FULL
1 df_raw.write.option("overwriteSchema",true).mode("overwrite").format('parquet').saveAsTable(delta_table_name, path = deltaFile)
(2) Spark Jobs
Command took 23.55 seconds -- by victornunesvidal@hotmail.com at 01/10/2023, 16:39:38 on VictorCluster

Cmd 5
```

Não seria necessário criar a tabela Gold agora, mas criei a mesma, a fim de facilitar futuras análises.

ETAPA 15 – ARQUITETURA





OBRIGADO

VICTORNUNESVIDAL@HOTMAIL.COM