

 MVP - Engenharia de dados

 Engenharia de Dados para análise de Jogos de Tabuleiro

Objetivo:

Este projeto tem como objetivo explorar, analisar e aplicar princípios de engenharia de dados em um dataset contendo informações sobre avaliações de jogos de tabuleiro. A execução seguirá etapas essenciais, tais como: ingestão de dados, limpeza e transformação dos dados, modelagem, armazenamento e organização, disponibilização e, por fim, análise e elaboração de relatórios.

O desafio central consiste em transformar dados dispersos em conhecimento estruturado e extrair resultados e insights significativos por meio da resposta a questionamentos analíticos, de modo a identificar tendências, fatores de popularidade, padrões de comportamento e relações entre jogos e jogadores na comunidade.

 Dataset - Coleta e Carga dos dados:

O conjunto de dados foi obtido na plataforma Kaggle.

[fonte: <https://www.kaggle.com/datasets/andrewmvb/board-games>] e reúne informações extraídas do site BoardGameGeek (BGG), com informações detalhadas sobre jogos, avaliações, mecânicas, categorias e perfis de jogadores, uma das maiores comunidades online dedicadas a jogos de tabuleiro.

O dataset foi hospedado no GitHub, sendo o mesmo espelhado no Databricks para carregamento e leitura dos dados, conforme mostra o código no [Notebook-MVP-Eng-Dados-Tabuleiro](#).

Just now (21s) 4: Coleta e Carga de Dados - Obtendo do workspace Python ⚡ []

```
# Coleta e carga de dados

# URL do dataset
file_path = "/Workspace/Users/marcio@pug@gmail.com/mvp-engenharia-dados/bgg_dataset.csv"

# Ler direto com pandas
df_dados = pd.read_csv(file_path, delimiter=';')

# Converter para Spark DataFrame
df_tabuleiro = spark.createDataFrame(df_dados)

# Informações do dataset carregado
print("Dataset carregado com sucesso!")
print(f"Total de registros: {df_tabuleiro.count()}")
print(f"Total de colunas..: {len(df_tabuleiro.columns)}")

> See performance (1) Optimize
```

df_dados: pandas.core.frame.DataFrame = [ID: float64, Name: object ... 12 more fields]
df_tabuleiro: pyspark.sql.connect.dataframe.DataFrame = [ID: double, Name: string ... 12 more fields]

Dataset carregado com sucesso!
Total de registros: 20343
Total de colunas..: 14

Pré-visualização dos dados:

08:34 PM (10)

6: Visualizar amostra dos dados

Python | See performance (1)

Table +

1.2 ID	A% Name	1.2 Year Pub...	A% Min Playa...	A% Max Playa...	A% Play Time	A% Min Age	A% Users R...	A% Rating Ave...	A% BGG Ra...	A% Completa...	1.2 Owned ...	A% Mechanics	A% Domains
1	174430	Gloomhaven	2017	1	4	120	14	42055	8.79	1	3.86	68323	> Action Queue, Action Retrieval, Campai...
2	161936	Pandemic Legacy: Season 1	2015	2	4	60	13	41643	8.61	2	2.84	65294	> Action Points, Cooperative Game, Hand...
3	224517	Brass: Birmingham	2018	2	4	120	14	19217	8.66	3	3.91	28785	> Hand Management, Income, Loans, Ma...
4	167791	Terrafirma Mars	2016	1	5	120	12	46464	8.43	4	3.24	87099	> Card Drafting, Drafting, End Game Bon...
5	233078	Twilight Imperium: Fourth Edition	2017	3	6	480	14	13468	8.7	5	4.22	16831	> Action Drafting, Area Majority / Influ...

5 rows | 1.50s runtime

Refreshed 18 minutes ago

As etapas de tratamentos de inconsistências (converter colunas de decimal para inteiro, substituir vírgulas por ponto em decimais e verificar valores ausentes) estão descritas no código localizado no notebook:

Notebook-MVP-Eng-Dados-Tabuleiro.

```
# Mostrar as primeiras 5 linhas
df_tabuleiro.select("ID", "Year Published", "Rating Average", "Complexity Average", "Owned Users").show(5)
```

ID	Year Published	Rating Average	Complexity Average	Owned Users
174430.0	2017.0	8,79	3,86	68323.0
161936.0	2015.0	8,61	2,84	65294.0
224517.0	2018.0	8,66	3,91	28785.0
167791.0	2016.0	8,43	3,24	87099.0
233078.0	2017.0	8,7	4,22	16831.0

only showing top 5 rows

Antes:

```
# Visualizar amostra dos dados
df_tabuleiro.select("ID", "Year Published", "Rating Average", "Complexity Average", "Owned Users").show(5)
```

ID	Year Published	Rating Average	Complexity Average	Owned Users
174430	2017	8.79	3.86	68323
161936	2015	8.61	2.84	65294
224517	2018	8.66	3.91	28785
167791	2016	8.43	3.24	87099
233078	2017	8.7	4.22	16831

only showing top 5 rows

Depois:

Dados curados sem valores ausentes.

```
df_resumo.show(truncate=False)
> See performance (29)

> df_tabuleiro: pyspark.sql.connect.DataFrame = [ID: integer, Name: string ... 12 more fields]
> df_resumo: pyspark.sql.connect.DataFrame = [Atributos: string, Valores Presentes: long ... 1 more field]

+-----+-----+-----+
|Atributos |Valores Presentes|Valores Ausentes|
+-----+-----+-----+
|ID      |9709    |0     |
|Name    |9709    |0     |
|Year Published |9709   |0     |
|Min Players |9709    |0     |
|Max Players |9709    |0     |
|Play Time |9709    |0     |
|Min Age   |9709    |0     |
|Users Rated|9709    |0     |
|Rating Average|9709   |0     |
|BGG Rank  |9709    |0     |
|Complexity Average|9709  |0     |
|Owned Users|9709    |0     |
|Mechanics  |9709    |0     |
|Domains   |9709    |0     |
+-----+-----+-----+
```

Valores mínimo, máximo e valores únicos, antes dos atributos serem traduzidos.

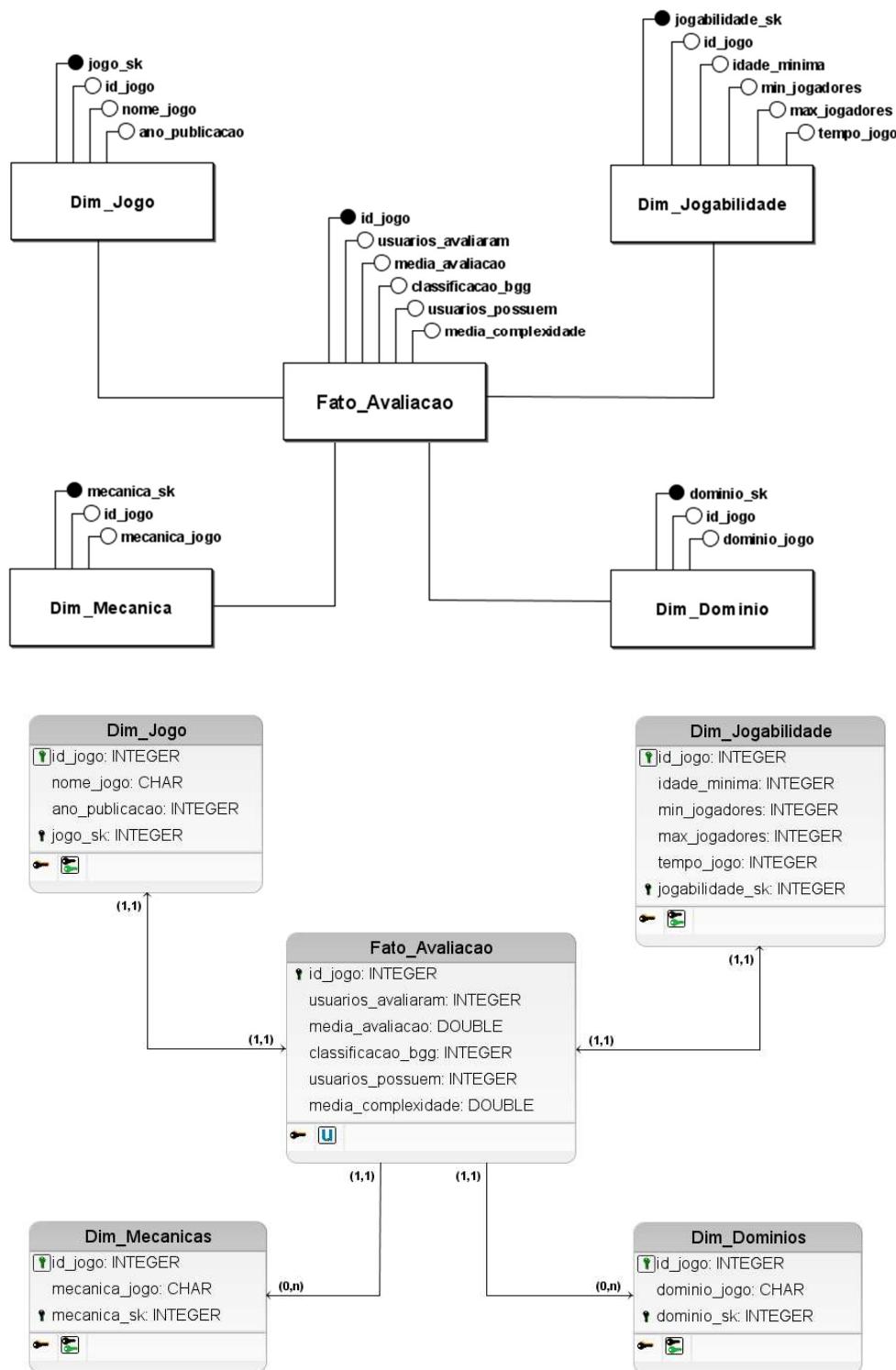
```
range_valores.show(truncate=False)
> See performance (1)

+-----+-----+-----+
|Atributos |Mínimo|Máximo|Qtd Valores Únicos|
+-----+-----+-----+
|ID       |1      |322289|9709   |
|Name     |-     |-     |9592   |
|Year Published |-3500 |2021 |162    |
|Min Players |0     |10    |10     |
|Max Players |0     |163   |36     |
|Play Time |0     |60000|99     |
|Min Age   |0     |21    |19     |
|Users Rated|30    |102214|2935  |
|Rating Average|1.43 |9.34  |522    |
|BGG Rank  |1     |20344 |9709   |
|Complexity Average|0.0  |5.0   |377    |
|Owned Users|3     |155312|3821  |
|Mechanics  |-     |-     |4589   |
|Domains   |-     |-     |39     |
+-----+-----+-----+
```

Os atributos e seus respectivos dados foram modelados para alimentar o modelo estrela (tabela fato e tabelas dimensão) durante a criação das tabelas fato e dimensão, conforme descrito no código localizado no notebook: [Notebook-MVP-Eng-Dados-Tabuleiro](#).

🔗 Modelagem

Através das informações do dataset, foi realizado a modelagem em estrela com tabelas com fato e dimensões. A seguir é mostrado o **modelo conceitual** e **lógico** do modelo estrela respectivamente.



Tabelas do Modelo Estrela

É uma estrutura de banco de dados otimizada para consultas analíticas e relatórios, em contraste com os modelos normalizados (OLTP) otimizados para transações. O seu nome vem da sua forma visual, que se assemelha a uma estrela: uma tabela central (fatos) cercada por várias tabelas auxiliares (dimensões).

Chave Substituta (Surrogate Key) é uma chave artificial, numérica e sequencial criada especificamente para o data warehouse, que substitui as chaves naturais (business keys) dos sistemas operacionais. Criada apenas para conectar tabelas de forma eficiente no data warehouse.

Tabelas Dimensão

São as "pontas" da estrela. Elas descrevem o contexto dos fatos.

Contêm os atributos descritivos e textuais usados para filtrar, agrupar e rotular os dados.

Criação Tabela dimensão Jogo - `dim_jogo`:

```
▶ ✅ 08:58 PM (1s) 18: Tabela Dimensão Jogo
# Criação Tabela dimensão Jogo
dim_jogo = df_tabuleiro.select(
    col("ID").alias("id_jogo"),
    col("Name").alias("nome_jogo"),
    col("Year Published").alias("ano_publicacao"),
).distinct()

# Adicionar chave substituta
dim_jogo = dim_jogo.withColumn("jogo_sk", monotonically_increasing_id())

# Visualiza amostra da tabela
print("Dimensão Jogo:")
display(dim_jogo.limit(5))
> See performance (1)

> dim_jogo: pyspark.sql.connect.DataFrame = [id_jogo: integer, nome_jogo: string ... 2 more fields]
Dimensão Jogo:
```

	id_jogo	nome_jogo	ano_publicacao	jogo_sk
1	146508	T.I.M.E Stories	2015	0
2	234477	Battle for Rokugan	2017	1
3	216091	Unlock! Escape Adventures - The Form...	2017	2
4	286096	Tapestry	2019	3
5	155362	Ca\$h' n Guns (Second Edition)	2014	4

↓ ↓ 5 rows | 1.26s runtime

Criação Tabela dimensão Jogabilidade - `dim_jogabilidade`:

```
▶ ✅ 08:59 PM (1s) 19: Tabela Dimensão Jogabilidade
# Criação Tabela dimensão Jogabilidade
dim_jogabilidade = df_tabuleiro.select(
    col("ID").alias("id_jogo"),
    col("Min Players").alias("min_jogadores"),
    col("Max Players").alias("max_jogadores"),
    col("Play Time").alias("tempo_jogo"),
    col("Min Age").alias("idade_minima")
).distinct()

# Adicionar chave substituta
dim_jogabilidade = dim_jogabilidade.withColumn("jogabilidade_sk", monotonically_increasing_id())

# Visualiza amostra da tabela
print("Dimensão Jogabilidade:")
display(dim_jogabilidade.limit(5))
> See performance (1)

> dim_jogabilidade: pyspark.sql.connect.DataFrame = [id_jogo: integer, min_jogadores: integer ... 4 more fields]
Dimensão Jogabilidade:
```

	id_jogo	min_jogadores	max_jogadores	tempo_jogo	idade_minima	jogabilidade_sk
1	156689	1	5	45	14	0
2	198773	2	8	15	10	1
3	2507	3	6	90	14	2
4	216132	1	4	120	12	3
5	152162	2	6	30	8	4

↓ ↓ 5 rows | 1.05s runtime

Criação Tabela dimensão Mecânicas - `dim_mecanicas`:

```
▶ ✓ 08:59 PM (1s) 20: Tabela Dimensão Mecânicas

# Criação Tabela dimensão mecanicas
dim_mecanicas = df_tabuleiro.select(
    col("ID").alias("id_jogo"),
    explode(split(col("Mechanics"), ",")).alias("mecanica_jogo")
).distinct()

# Adicionar chave substituta
dim_mecanicas = dim_mecanicas.withColumn("mecanica_sk", monotonically_increasing_id())

# Visualiza amostra da tabela
print("Dimensão Mecânicas:")
display(dim_mecanicas.limit(5))
> See performance (1)

dim_mecanicas: pyspark.sql.connect.DataFrame = [id_jogo: integer, mechanica_jogo: string ... 1 more field]

Dimensão Mecânicas:
```

	² ₃ id_jogo	^A _C mechanica_jogo	² ₃ mechanica_sk
1	171623	Income	0
2	150376	Variable Player Powers	1
3	54998	Auction: Fixed Placement	2
4	191862	Set Collection	3
5	12942	Auction/Bidding	4

5 rows | 0.95s runtime

Criação Tabela dimensão Domínios - `dim_dominios`:

```
▶ ✓ 08:59 PM (1s) 21: Tabela Dimensão Domínios

# Criação Tabela dimensão Domínios
dim_dominios = df_tabuleiro.select(
    col("ID").alias("id_jogo"),
    explode(split(col("Domains"), ",")).alias("dominio_jogo")
).distinct()

# Adicionar chave substituta
dim_dominios = dim_dominios.withColumn("dominio_sk", monotonically_increasing_id())

# Visualiza amostra da tabela
print("Dimensão Domínios:")
display(dim_dominios.limit(5))
> See performance (1)

dim_dominios: pyspark.sql.connect.DataFrame = [id_jogo: integer, dominio_jogo: string ... 1 more field]

Dimensão Domínios:
```

	² ₃ id_jogo	^A _C dominio_jogo	² ₃ dominio_sk
1	125548	Strategy Games	0
2	271518	Family Games	1
3	113873	Wargames	2
4	220877	Strategy Games	3
5	235488	Family Games	4

5 rows | 0.96s runtime

Tabelas Fato

É o "coração" do modelo. Representa o evento ou processo de negócio que se deseja analisar.

Contém as medidas ou métricas numéricas.

É composta principalmente por chaves estrangeiras (FK) que se conectam às tabelas de dimensão e pelas medidas em si.

Criação Tabela Fato Avaliacao - fato_avaliacao:

```

▶ ✓ 08:59 PM (1s)                                     23: Tabela Fato Avaliação

# Criação Tabela Fato Avaliacao
fato_avaliacao = df_tabuleiro.select(
    col("ID").alias("id_jogo"),
    col("Users Rated").alias("usuarios_avaliaram"),
    col("Rating Average").alias("media_avaliacao"),
    col("BGG Rank").alias("classificacao_bg"),
    col("Complexity Average").alias("media_complexidade"),
    col("Owned Users").alias("usuarios_posuem")
)

# Visualiza amostra da tabela
print("Tabela Fato Avaliação:")
display(fato_avaliacao.limit(5))
> See performance (1)

> fato_avaliacao: pyspark.sql.connect.DataFrame = [id_jogo: integer, usuarios_avaliaram: integer ... 4 more fields]

```

Tabela Fato Avaliação:

Table						
	id_jogo	usuarios_avaliaram	media_avaliacao	classificacao_bg	media_complexidade	usuarios_posuem
1	174430	42055	8.79	1	3.86	68323
2	161936	41643	8.61	2	2.84	65294
3	224517	19217	8.66	3	3.91	28785
4	167791	64864	8.43	4	3.24	87099
5	233078	13468	8.7	5	4.22	16831

5 rows | 0.73s runtime

Catálogo de Dados

Catalog

∅ Serverless Starter Warehouse Serverless 2XS

Type to search...

My organization

- workspace
 - default
 - information_schema
- tabuleiro
 - dim_dominios
 - dim_jogabilidade
 - dim_jogo
 - dim_mecanicas
 - fato_avaliacao

Catalog: workspace

Database ou Schema: tabuleiro

tabela dimensão:
dim_dominios; dim_jogabilidade; dim_jogo; dim_mecanicas

tabela fato: fato_avaliacao

Descrição dos atributos:

tabuleiro					
TABELA	ATRIBUTO (original)	ATRIBUTO (traduzido)	TIPO	VALORES	COMENTÁRIO
dim_jogo	ID	id_jogo	int	1 a 3222289	ID original do jogo no dataset BoardGameGeek (BGG) (1 a 3222289)
	Name	nome_jogo	string	-	Nome completo e oficial do jogo de tabuleiro
	Year Published	ano_publicacao	int	-3500 a 2021	Ano de publicação original do jogo (-3500 a 2021)
	-	jogo_sk	bigint	-	Chave substituta artificial para a dimensão jogo
dim_jogabilidade	ID	id_jogo	int	1 a 3222289	ID original do jogo no dataset BoardGameGeek (BGG) (1 a 3222289)
	Min Players	min_jogadores	int	0 a 10	Número mínimo de jogadores necessários (0 a 10)
	Max Players	max_jogadores	int	0 a 163	Número máximo de jogadores suportados (0 a 163)
	Play Time	tempo_jogo	int	0 a 60000	Duração média da partida em minutos (0 a 60000)
	Min Age	idade_minima	int	0 a 21	Idade mínima recomendada para jogar (0 a 21)
	-	jogabilidade_sk	bigint	-	Chave substituta artificial para a dimensão jogabilidade
dim_dominios	ID	id_jogo	int	1 a 3222289	ID original do jogo no dataset BoardGameGeek (BGG) (1 a 3222289)
	Domains	dominio_jogo	string	-	Categorias temáticas ou gêneros dos jogos.
	-	dominio_sk	bigint	-	Chave substituta artificial para a dimensão domínios
dim_mecanicas	ID	id_jogo	int	1 a 3222289	ID original do jogo no dataset BoardGameGeek (BGG) (1 a 3222289)
	Mechanics	mecanica_jogo	string	-	Regras e dinâmica de jogabilidade. Como o jogo funciona.
	-	mecanica_sk	bigint	-	Chave substituta artificial para a dimensão mecânicas
fato_avaliacao	ID	id_jogo	int	1 a 3222289	ID original do jogo no dataset BoardGameGeek (BGG) (1 a 3222289)
	Users Rated	usuarios_avaliaram	int	30 a 102214	Quantidade de usuários que avaliaram o jogo (30 a 102214)
	Rating Average	media_avaliacao	double	1.43 a 9.34	Nota média recebida (1.43 a 9.34)
	BGG Rank	classificacao_bgg	int	1 a 20344	Posição no ranking da BoardGameGeek (1 a 20344)
	Complexity Average	media_complexidade	double	0 a 5	Nível médio de dificuldade/complexidade (0 a 5)
	Owned Users	usuarios_posseuem	int	3 a 155312	Quantidade de usuários que possuem o jogo (3 a 155312)

Catalog Explorer > workspace > tabuleiro >

dim_jogo

Overview Sample Data Details Permissions History Lineage Insights Quality

Description

[AI generate](#) [Add](#)

Filter columns...				
Column	Type	Comment	Tags	Column masking
id_jogo	int	ID original do jogo no dataset BoardGameGeek (BGG) (1 a 3222289)	identificador	
nome_jogo	string	Nome completo e oficial do jogo de tabuleiro	descricao	
ano_publicacao	int	Ano de publicação original do jogo (-3500 a 2021)	temporal	
jogo_sk	bigint	Chave substituta artificial para a dimensão jogo	chave_substituta	

Catalog Explorer > workspace > tabuleiro >

dim_jogabilidade [Overview](#) [Sample Data](#) [Details](#) [Permissions](#) [Policies](#) [History](#) [Lineage](#) [Insights](#) [Quality](#)

Description

[Add](#) Filter columns...

Column	Type	Comment	Tags	Column masking
id_jogo	int	ID original do jogo no dataset BoardGameGeek (BGG)	identificador	
min_jogadores	int	Número mínimo de jogadores necessários (0 a 10)	jogabilidade_min	
max_jogadores	int	Número máximo de jogadores suportados (0 a 163)	jogabilidade_max	
tempo_jogo	int	Duração média da partida em minutos (0 a 60000)	duracao	
idade_minima	int	Idade mínima recomendada para jogar (0 a 21)	faixa_etaria	
jogabilidade_sk	bigint	Chave substituta artificial para a dimensão jogabilidade	chave_substituta	

Catalog Explorer > workspace > tabuleiro >

dim_dominios [Overview](#) [Sample Data](#) [Details](#) [Permissions](#) [Policies](#) [History](#) [Lineage](#) [Insights](#) [Quality](#)

Description

[Add](#) Filter columns...

Column	Type	Comment	Tags	Column masking
id_jogo	int	ID original do jogo no dataset BoardGameGeek (BGG)	identificador	
dominio_jogo	string	Categorias temáticas ou gêneros dos jogos. O que o jogo é (categoria/tema).	categoria	
dominio_sk	bigint	Chave substituta artificial para a dimensão domínios	chave_substituta	

Catalog Explorer > workspace > tabuleiro >

dim_mecanicas [Overview](#) [Sample Data](#) [Details](#) [Permissions](#) [Policies](#) [History](#) [Lineage](#) [Insights](#) [Quality](#)

Description

[Add](#) Filter columns...

Column	Type	Comment	Tags	Column masking
id_jogo	int	ID original do jogo no dataset BoardGameGeek (BGG)	identificador	
mecanica_jogo	string	Regras e dinâmica de jogabilidade. Como o jogo funciona (regras/dinâmica).	dinamica	
mecanica_sk	bigint	Chave substituta artificial para a dimensão mecânicas	chave_substituta	

Catalog Explorer > workspace > tabuleiro >

fato_avaliacao

Overview Sample Data Details Permissions History Lineage Insights Quality

Description

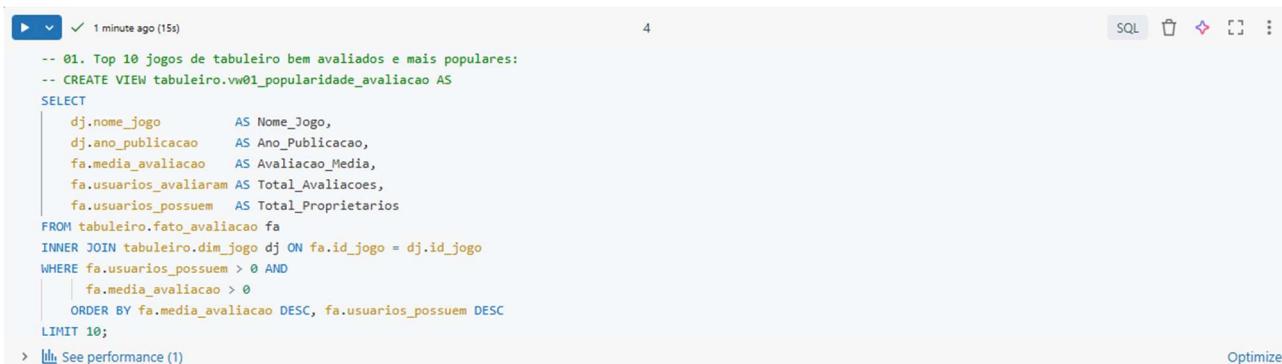
Filter columns...

Column	Type	Comment	Tags	Column masking
id_jogo	int	ID original do jogo no dataset BoardGameGeek (BGG)	identificador	
usuarios_avaliaram	int	Quantidade de usuários que avaliaram o jogo (30 a 102214)	engajamento	
media_avaliacao	double	Nota média recebida (1.43 a 9.34)	desempenho	
classificacao_bgg	int	Posição no ranking da BoardGameGeek (1 a 20344)	ranking	
media_complexidade	double	Nível médio de dificuldade/complexidade (0 a 5)	nível	
usuarios_posuem	int	Quantidade de usuários que possuem o jogo (3 a 155312)	popularidade	

QUESTÕES ANALÍTICAS RESPONDIDAS.

As queries utilizadas estão registradas no notebook **Query-Tabuleiro-Questoes-Solucoes.ipynb** (<https://github.com/marcioipugnal/mvp-engenharia-dados/blob/main/Query-Tabuleiro-Questoes-Solucoes.ipynb>), o qual está hospedado na mesma área do repositório do GitHub deste relatório.

1. Quais são os jogos (top 10) de tabuleiro mais bem avaliados e sua relação com a popularidade?

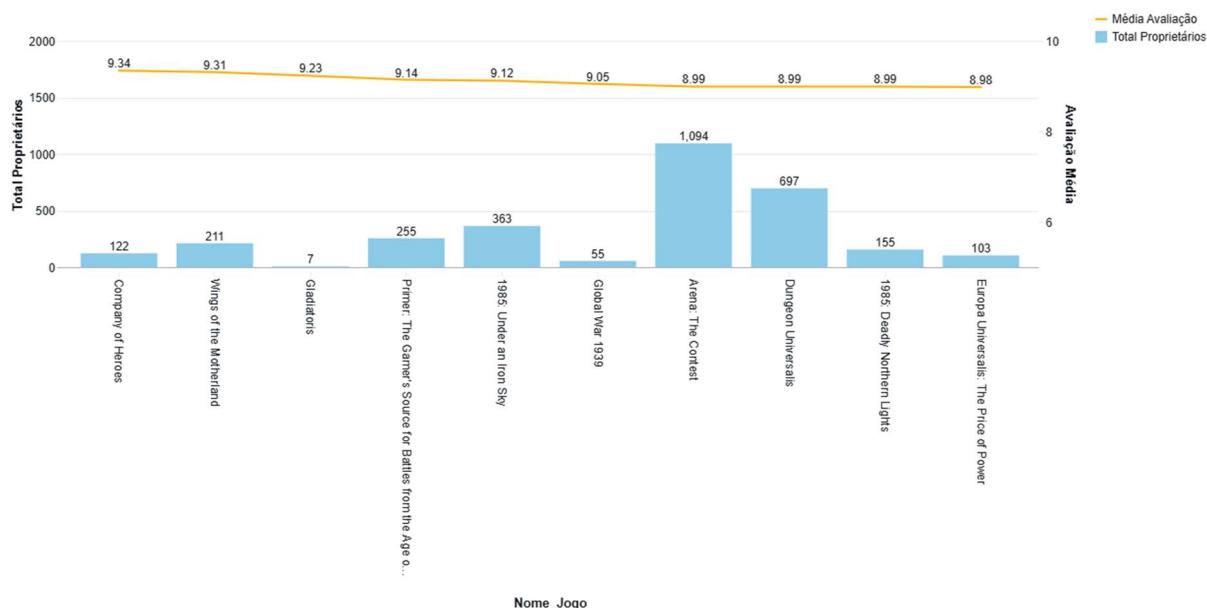


```
-- 01. Top 10 jogos de tabuleiro bem avaliados e mais populares:
-- CREATE VIEW tabuleiro.vw01_popularidade_avaliacao AS
SELECT
    dj.nome_jogo          AS Nome_Jogo,
    dj.ano_publicacao     AS Ano_Publicacao,
    fa.media_avaliacao   AS Avaliacao_Media,
    fa.usuarios_avaliaram AS Total_Avaliacoes,
    fa.usuarios_posseum  AS Total_Proprietarios
FROM tabuleiro.fato_avaliacao fa
INNER JOIN tabuleiro.dim_jogo dj ON fa.id_jogo = dj.id_jogo
WHERE fa.usuarios_posseum > 0 AND
      fa.media_avaliacao > 0
ORDER BY fa.media_avaliacao DESC, fa.usuarios_posseum DESC
LIMIT 10;
```

SQL    

See performance (1) Optimize

Table		Jogo vs Avaliação	Jogo vs Popularidade	Avaliação vs Popularidade	+
		¹ Nome_Jogo	² Ano_Publicacao	^{1,2} Avaliacao_Media	^{1,3} Total_Avaliacoes
1	Company of Heroes		2020	9.34	47
2	Wings of the Motherland		2019	9.31	79
3	Gladiators		2009	9.23	31
4	Primer: The Gamer's Source for Battles from the Age of Reas...		2013	9.14	58
5	1985: Under an Iron Sky		2018	9.12	90
6	Global War 1939		2011	9.05	34
7	Arena: The Contest		2019	8.99	600
8	Dungeon Universalis		2019	8.99	454
9	1985: Deadly Northern Lights		2020	8.99	36
10	Europa Universalis: The Price of Power		2021	8.98	74



Comentário:

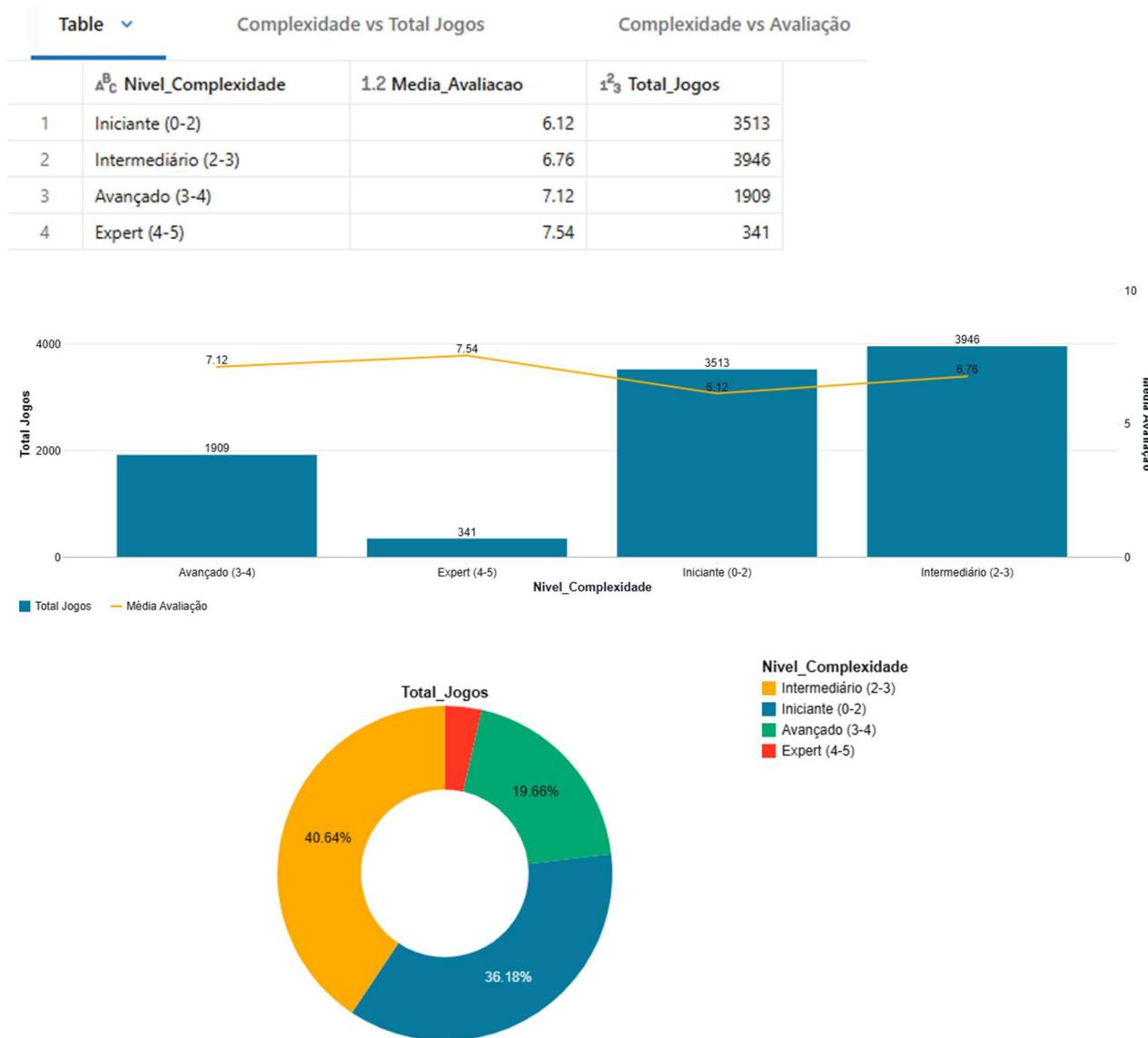
- Os jogos mais bem avaliados não são necessariamente os mais populares.
- A maioria dos jogos com notas altas é relativamente recente (2018–2021).
- Jogos com muitas avaliações tendem a ter notas ligeiramente menores, mas mais confiáveis estatisticamente.
- Jogos com poucas avaliações podem inflar notas, mas não representam consenso estatístico.
- Jogos com um volume maior de avaliações tendem a ter notas ligeiramente mais moderadas, porém estatisticamente mais confiáveis, refletindo um consenso mais amplo.

2. Como o nível de complexidade dos jogos influencia a avaliação média dos jogos?

```

▶ ▾ ✓ 10:02 AM (2s) 7 SQL 🗑 ⚙️ ⌂ ⋮
-- 02. Nível de complexidade por média de avaliação dos jogos
-- CREATE VIEW tabuleiro.vw02_complexidade_avaliacao AS
SELECT
CASE
    WHEN fa.media_complexidade < 2 THEN 'Iniciante (0-2)'
    WHEN fa.media_complexidade < 3 THEN 'Intermediário (2-3)'
    WHEN fa.media_complexidade < 4 THEN 'Avançado (3-4)'
    ELSE 'Expert (4-5)'
END AS Nivel_Complexidade,
ROUND(AVG(fa.media_avaliacao), 2) AS Media_Avaliacao,
COUNT(*) AS Total_Jogos
FROM tabuleiro.fato_avaliacao fa
WHERE fa.media_complexidade IS NOT NULL
GROUP BY Nivel_Complexidade
ORDER BY Media_Avaliacao ASC;
> See performance (!) Optimize

```



💡 Comentário:

- Jogos mais complexos (avançado e expert) tendem a receber avaliações mais altas. Há uma clara correlação positiva: quanto maior a complexidade, maior a média de avaliação.
- A maior parte dos jogos está concentrada nos níveis Iniciante e Intermediário (mais de 7.400 títulos).
- Jogos simples são abundantes, mas não tão bem avaliados.
- Jogos complexos são menos numerosos, mas recebem notas mais altas.

3. Quais as mecânicas de jogos que estão associadas às maiores avaliações e popularidade?

10 SQL ⚙️ ⚡ ⚡ ⚡ ⚡ ⚡

```
-- 03. Mecânicas mais bem avaliadas e sua popularidade.
-- CREATE VIEW tabuleiro.vw03_mecanicas_avaliadas_populares AS
SELECT
    dm.mecanica_jogo AS Mecanica_Jogo,
    ROUND(AVG(fa.media_avaliacao), 2) AS Avaliacao_Media,
    SUM(fa.usuarios_posseum) AS Popularidade,
    COUNT(DISTINCT fa.id_jogo) AS Total_Jogos
FROM tabuleiro.fato_avaliacao fa
JOIN tabuleiro.dim_mecanicas dm
ON fa.id_jogo = dm.id_jogo
GROUP BY dm.mecanica_jogo
ORDER BY Avaliacao_Media DESC, Popularidade DESC
LIMIT 10;
> See performance (1) Optimize
```

Table Mecânicas Jogo vs Avaliação Mecânicas Jogo vs Popularidade

	Mecanica_Jogo	Avaliacao_Media	Popularidade	Total_Jogos
1	Legacy Game	7.87	280037	20
2	Automatic Resource Grow...	7.82	228094	11
3	Ownership	7.78	311224	27
4	Delayed Purchase	7.77	258303	10
5	Turn Order: Pass Order	7.75	176496	12
6	Turn Order: Auction	7.74	146152	10
7	Turn Order: Claim Action	7.7	474439	28
8	Victory Points as a Resource	7.68	476511	36
9	Command Cards	7.68	93200	10
10	Market	7.66	322978	35



Comentário:

- As mecânicas listadas têm sua avaliação próxima, sem grandes discrepâncias, o que indica homogeneidade na avaliação.
- Não há mecânicas com notas muito baixas ou muito altas.
- As mecânicas mais populares não são necessariamente as mais bem avaliadas.
- Popularidade não garante avaliação máxima.

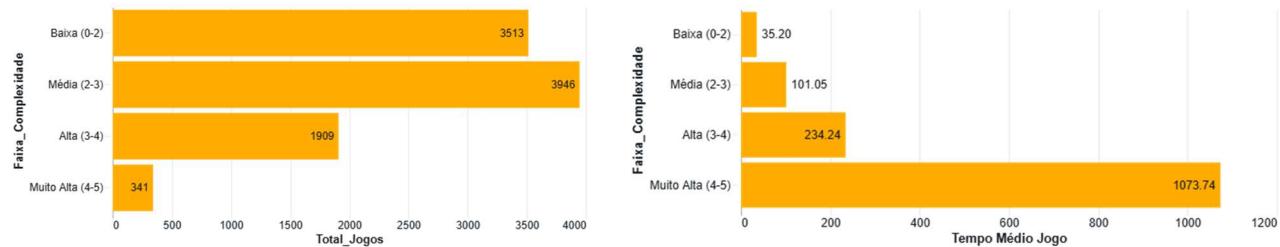
4. Qual a correlação entre a complexidade dos jogos e o tempo médio de uma partida?

```

▶ ▾ ✓ 10:02 AM (2s) 13 SQL ⌂ ⌄ ⌅ ⌆
-- 04. Complexidade média e tempo de jogo
-- CREATE VIEW tabuleiro.vw04_complexidade_tempo_jogo AS
SELECT
CASE
    WHEN fa.media_complexidade < 2 THEN 'Baixa (0-2)'
    WHEN fa.media_complexidade < 3 THEN 'Média (2-3)'
    WHEN fa.media_complexidade < 4 THEN 'Alta (3-4)'
    ELSE 'Muito Alta (4-5)'
END AS Faixa_Complexidade,
ROUND(AVG(djogab.tempo_jogo), 2) AS Tempo_Medio_Jogo,
ROUND(AVG(fa.media_avaliacao), 2) AS Avaliacao_Media,
COUNT(DISTINCT fa.id_jogo) AS Total_Jogos
FROM tabuleiro.fato_avaliacao fa
JOIN tabuleiro.dim_jogabilidade djogab
ON fa.id_jogo = djogab.id_jogo
GROUP BY Faixa_Complexidade
ORDER BY Tempo_Medio_Jogo ASC;
> See performance (1) Optimize

```

	Faixa_Complexidade	1.2 Tempo_Medio_Jogo	1.2 Avaliacao_Media	1.3 Total_Jogos
1	Baixa (0-2)	35.2	6.12	3513
2	Média (2-3)	101.05	6.76	3946
3	Alta (3-4)	234.24	7.12	1909
4	Muito Alta (4-5)	1073.74	7.54	341



Comentário:

- Jogos com baixa e média complexidade são jogos rápidos, mas esses não alcançam notas tão altas.
- Têm acesso a uma grande oferta de jogos rápidos e simples, mas esses não alcançam notas tão altas.
- Jogos de muito alta complexidade são raros (apenas 341), mas recebem a melhor avaliação média.
- Jogadores tendem a valorizar mais jogos com alta complexidade, pode ser por ser mais desafiador e profundo, mesmo que exijam mais tempo.

5. Como a faixa etária recomendada pelo jogo influencia a avaliação dos jogos?

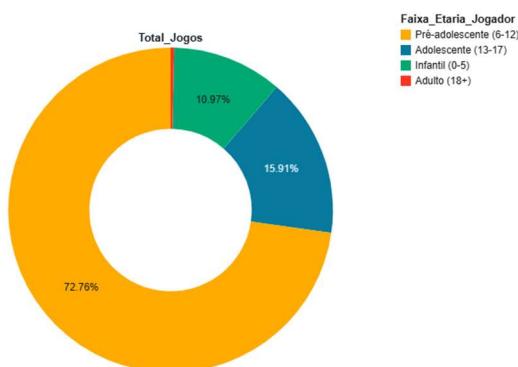
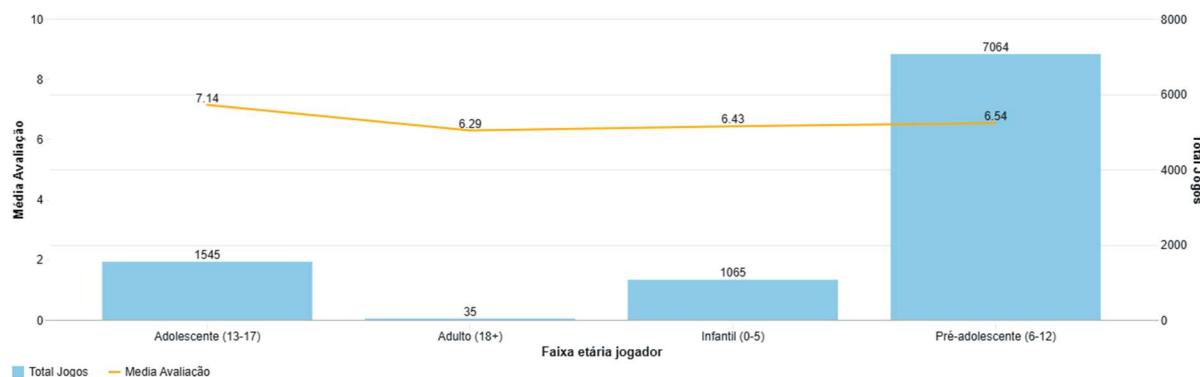
10:02 AM (3s) 16 SQL ⚙️ ⚡ ⚡ ⚡

```
-- 05. Comparação de avaliação por faixa etária recomendada
-- CREATE VIEW tabuleiro.vw05_avaliacao_faixa_etaria AS
SELECT
CASE
    WHEN djg.idade_minima BETWEEN 0 AND 5 THEN 'Infantil (0-5)'
    WHEN djg.idade_minima BETWEEN 6 AND 12 THEN 'Pré-adolescente (6-12)'
    WHEN djg.idade_minima BETWEEN 13 AND 17 THEN 'Adolescente (13-17)'
    ELSE 'Adulto (18+)'
END AS Faixa_Etaria_Jogador,
ROUND(AVG(fa.media_avaliacao), 2) AS Media_Avaliacao_Jogo,
COUNT(DISTINCT fa.id_jogo) AS Total_Jogos
FROM tabuleiro.fato_avaliacao fa
JOIN tabuleiro.dim_jogabilidade djg
ON fa.id_jogo = djg.id_jogo
GROUP BY Faixa_Etaria_Jogador
ORDER BY Media_Avaliacao_Jogo DESC;
```

> See performance (1) Optimize

Table Faixa etária vs Total Jogos Faixa etária vs Média Avaliação

	A. Faixa_Etaria_Jogador	B. Media_Avaliacao_Jogo	C. Total_Jogos
1	Adolescente (13-17)	7.14	1545
2	Pré-adolescente (6-12)	6.54	7064
3	Infantil (0-5)	6.43	1065
4	Adulto (18+)	6.29	35



Comentário:

- Melhor avaliação entre adolescentes (13-17).
- Adolescentes são o público que mais valoriza os jogos disponíveis.
- Maior oferta de jogos para pré-adolescentes (6-12).
- Mostra que há pouquíssimos jogos para adultos (18+) e avaliação não é tão alta, embora seja a menor nota, ela está relativamente próxima das demais faixas.

6. Quais categorias (domínios) temáticas tem maior popularidade e mantêm alta avaliação?



Comentário:

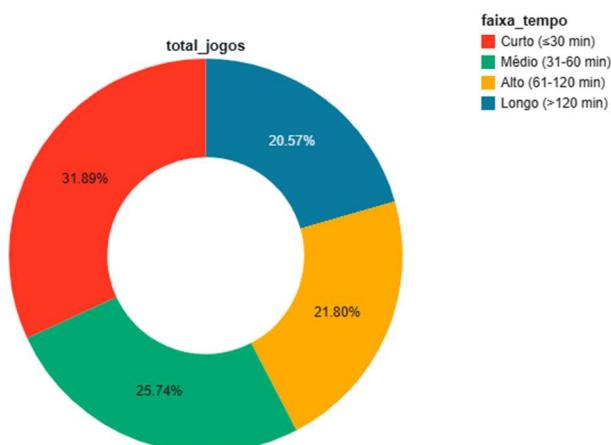
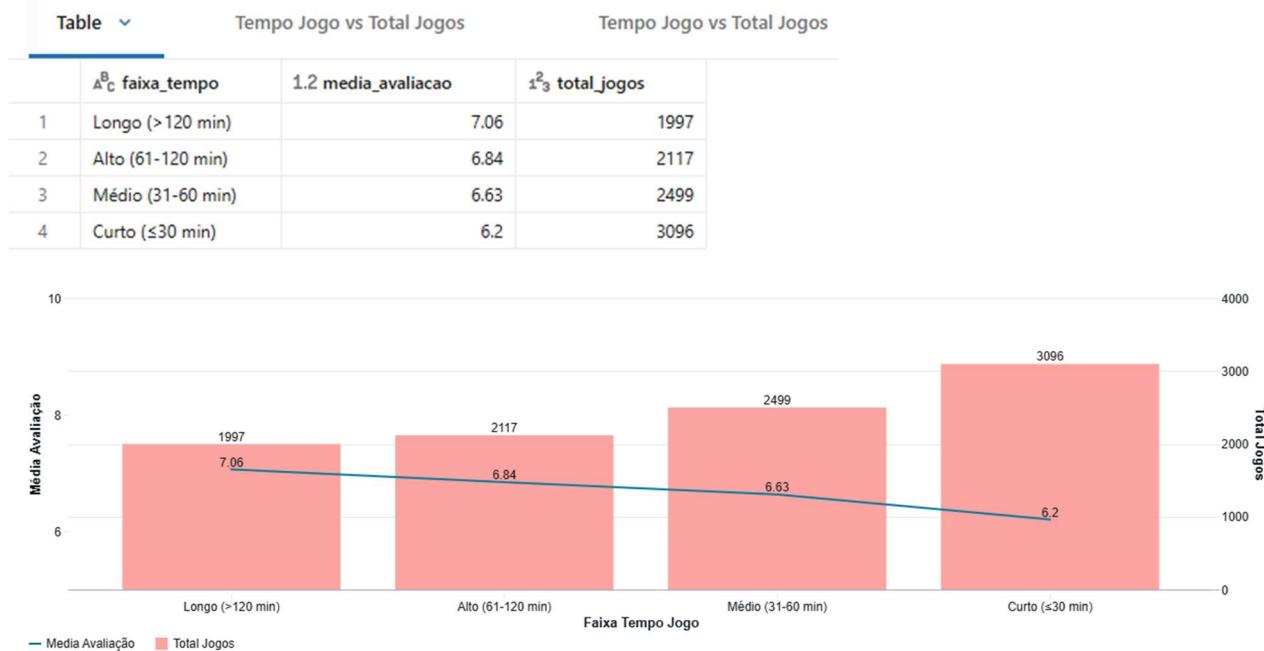
- *Strategy Games* - tem alta popularidade (9,7 milhões de proprietários) e têm boa avaliação (6.96), equilibrando alcance e qualidade.
- *Family Games* - também muito populares (8,7 milhões), mas com avaliação mediana (6.43), mostrando que volume não garante qualidade.
- *Wargames* - maior número de títulos (3.287), avaliação sólida (6.85), mas público menos massivo.
- *Thematic Games* - boa avaliação (6.81) e popularidade intermediária, valorizados por públicos específicos.
- *Customizable Games* - poucos títulos (252), mas comunidades fiéis (650 mil proprietários).
- *Party Games* e *Abstract Games* - populares em contextos sociais ou conceituais, mas com notas abaixo de 6.5.
- *Children's Games* - menor avaliação (5.53) e baixa popularidade, indicando espaço para inovação no segmento infantil.

7. Qual a relação entre tempo de jogo e satisfação dos jogadores?

```

▶ ▾ ✓ 10:57 PM (1s) 22 SQL 🗑 ⚡ ⌂ ⌂ ⌂
-- 07. Tempo de jogo vs. Avaliação
-- CREATE VIEW tabuleiro.vw07_tempo_jogo_avaliacao AS
SELECT
CASE
    WHEN j.tempo_jogo <= 30 THEN 'Curto (<30 min)'
    WHEN j.tempo_jogo BETWEEN 31 AND 60 THEN 'Médio (31-60 min)'
    WHEN j.tempo_jogo BETWEEN 61 AND 120 THEN 'Alto (61-120 min)'
    ELSE 'Longo (>120 min)'
END AS faixa_tempo,
ROUND(AVG(f.media_avaliacao),2) AS media_avaliacao,
COUNT(DISTINCT f.id_jogo) AS total_jogos
FROM tabuleiro.fato_avaliacao f
JOIN tabuleiro.dim_jogabilidade j ON f.id_jogo = j.id_jogo
GROUP BY faixa_tempo
ORDER BY media_avaliacao DESC;
> See performance (1)
Optimize

```



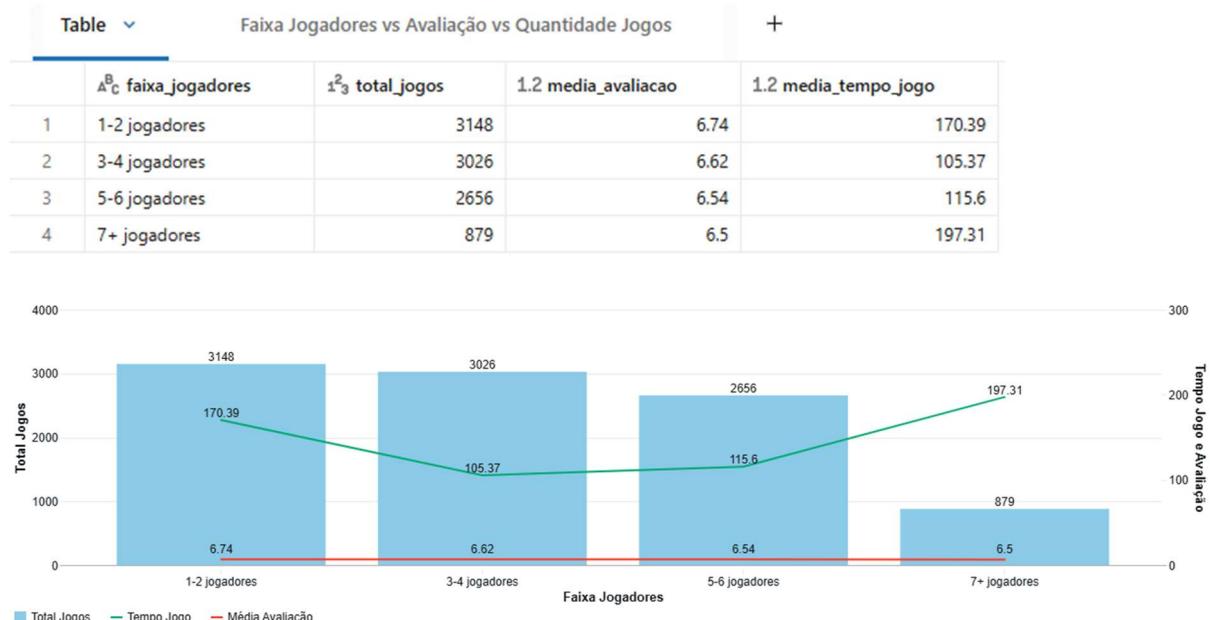
Comentário:

- Jogos longos (>120 min) tendem a ter notas mais altas.
- Jogos curtos (<30 min) são abundantes, mas menos valorizados.
- Satisfação cresce com maior investimento de tempo, sugerindo que jogadores associam duração a profundidade.

8. Qual é o perfil dos jogos (quantidade, avaliação e tempo) por faixa de jogadores?

10:57 PM (1s) 25 SQL 🗑️ ⚡ ☰ ⏪

```
-- 08. Perfil dos jogos por faixa de jogadores.
-- CREATE VIEW tabuleiro.vw08_perfil_faixa_jogadores AS
SELECT
CASE
    WHEN jog.max_jogadores <= 2 THEN '1-2 jogadores'
    WHEN jog.max_jogadores <= 4 THEN '3-4 jogadores'
    WHEN jog.max_jogadores <= 6 THEN '5-6 jogadores'
    ELSE '7+ jogadores'
END AS faixa_jogadores,
COUNT(fa.id_jogo) AS total_jogos,
ROUND(AVG(fa.media_avaliacao), 2) AS media_avaliacao,
ROUND(AVG(jog.tempo_jogo), 2) AS media_tempo_jogo
FROM
tabuleiro.fato_avaliacao fa JOIN
tabuleiro.dim_jogabilidade jog ON fa.id_jogo = jog.id_jogo
GROUP BY
faixa_jogadores
ORDER BY
total_jogos DESC;
> See performance (1) Optimize
```



💡 Comentário:

- Jogos para poucos jogadores são os mais bem avaliados.
- Verificado que a concentram em faixas de até 4 jogadores, que somam mais de 6.000 títulos.
- Grupos grandes (5+) têm menos opções, partidas longas e avaliações mais baixas.

✅ Conclusão

Este projeto demonstra como a engenharia de dados pode ser aplicada em ambientes analíticos para transformar informações dispersas em conhecimento estruturado e acionável.

Os resultados revelam que **popularidade não implica qualidade**: jogos amplamente jogados não são necessariamente os mais bem avaliados. Também foi possível identificar que **complexidade e duração** tendem a elevar as notas atribuídas pelos usuários, enquanto jogos curtos (≤ 30 minutos), apesar de numerosos, costumam receber avaliações mais modestas.

Além disso, surgem **lacunas relevantes para públicos específicos**, como adultos e grupos grandes, que contam com menor oferta de títulos e apresentam níveis inferiores de satisfação. As análises de categorias e mecânicas indicam a existência de nichos fiéis, mas mostram que isso **não garante**, por si só, avaliações consistentemente altas.

Em síntese, os achados reforçam que fatores como **complexidade, tempo de jogo e público-alvo** exercem influência significativa na percepção de qualidade dos jogos de tabuleiro, oferecendo insumos valiosos para decisões estratégicas.