

MVP – MBA em Ciência de Dados e Analytics

Módulo: Engenharia de Dados

Aluno: Leandro Martins Kobbi

Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)

Descrição do trabalho

Os Jogos Olímpicos representam um dos maiores eventos esportivos do mundo, reunindo atletas de diversas nações em competições de alto desempenho.

Dentro desse contexto, este trabalho busca explorar alguns dados disponíveis e realizar algumas análises sobre a história das Olimpíadas e os dados relacionados aos atletas que participaram ao longo dos anos.

Objetivo

O principal objetivo deste trabalho é **explorar as curiosidades e tendências dos Jogos Olímpicos** com base em dados históricos. A análise considera diversas dimensões, como o número de países participantes, o perfil dos atletas (idade, altura, peso), os esportes disputados e a distribuição de medalhas.

Para alcançar esses objetivos, serão realizadas **consultas e avaliações detalhadas** sobre as bases de dados disponíveis, buscando insights relevantes e tendências ao longo dos anos. Além disso, será explorada a **modelagem dos dados**, visando estruturar as informações de forma eficiente para futuras análises.

Aquisição dos Dados

Os arquivos utilizados neste trabalho foram obtidos manualmente a partir da plataforma **Kaggle**, por meio do seguinte link:

Fonte dos Dados: [120 Years of Olympic History: Athletes and Results]

<https://www.kaggle.com/datasets/heesoo37/120-years-ofolympic-history-athletes-and-results>

A base de dados consiste em **dois arquivos principais**, que contêm informações detalhadas sobre atletas e suas participações nos Jogos Olímpicos ao longo de 120 anos:

Base athlete_events.csv – Contém dados individuais dos atletas, incluindo nome, gênero, idade, altura, peso, equipe, país (NOC), ano, cidade-sede, esporte, evento e medalhas conquistadas.

Base noc_regions.csv – Mapeia os códigos dos Comitês Olímpicos Nacionais (NOC) para os respectivos países e regiões, permitindo relacionar as informações dos atletas com seus países de origem.

Perguntas a serem respondidas nas análises realizadas ao longo do trabalho

1. Identificação dos atletas mais pesados
2. Tem alguma relação idade e quantidade de medalhas
3. Qual a distribuição de idade dos atletas ao longo das edições dos Jogos Olímpicos?
4. Há uma relação entre a altura e o tipo de esporte praticado pelos atletas?
5. Como o peso médio dos atletas variou ao longo dos anos e em quais modalidades ele tende a ser mais elevado
6. Qual é a altura média dos medalhistas de ouro nos esportes disponíveis
7. Existe alguma relação esporte x quantidade de jogos
8. Atletas brasileiros com mais jogos

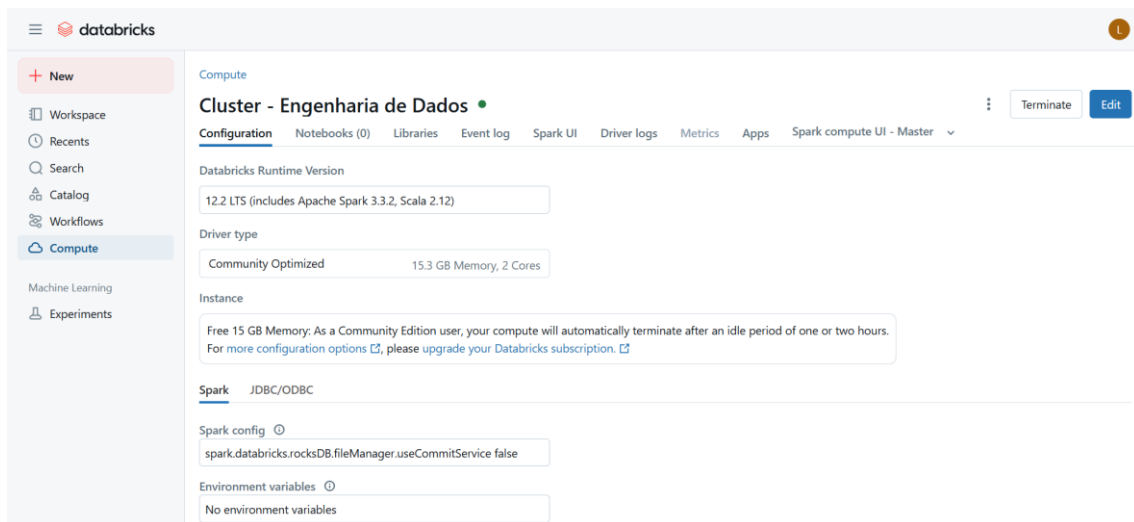
Glossário de dados:

ID	Coluna	Explicação
1	ID	Identificador único para cada atleta na tabela
2	Name	Nome do atleta
3	Sex	Gênero do atleta
4	Age	Idade do atleta durante os Jogos Olímpicos
5	Height	Altura do atleta em centímetros
6	Weight	Peso do atleta em quilogramas
7	Team	Nome do time ou nação representada pelo atleta
8	NOC	Código do Comitê Olímpico Nacional ao qual o atleta está associado
9	Games	Nome dos Jogos Olímpicos
10	Year	Ano em que os Jogos Olímpicos ocorreram
11	Season	Estação em que os Jogos Olímpicos ocorreram (verão ou inverno)
12	City	Nome da cidade onde os Jogos Olímpicos ocorreram
13	Sport	Nome do esporte em que o atleta competiu
14	Event	Nome específico do evento dentro do esporte
15	Medal	Medalha conquistada pelo atleta (ouro, prata, bronze ou nenhuma)

Passo a passo seguido:

Dentro do ambiente do Databricks, seguimos os seguintes passos para criar o cluster:

1. Acessar a aba "Cluster/ Computer" no menu lateral esquerdo.
2. Clicar no botão "Create Cluster/ Computer" (Criar Cluster).
3. Configurar as opções do cluster:
 - o Cluster Name: Nome escolhido para o cluster como "Cluster- Engenharia de Dados".
 - o Databricks Runtime Version: Escolhemos Databricks Runtime 12.2 LTS (Scala 2.12, Spark 3.3.2).
4. Clicar em "Create Cluster".



Verificação e Conexão ao Cluster

Após a criação do cluster, verificamos se ele estava "Running" (em execução) antes de iniciar a codificação em notebooks do Databricks.

Para garantir a conexão com o cluster:

- Acessar a aba "Clusters",
- Clicar no cluster "Cluster- Engenharia de Dados",
- Criar um novo notebook e o anexar ao cluster para rodar comandos em PySpark.

Criação do notebooks e execução dos códigos em PySpark

- Base Principal - athlete_events_1
- Base Auxiliar - noc_regions

Após a criação do cluster no **Databricks**, o próximo passo é subir notebooks para execução dos códigos em **PySpark**.

- 1- Clicar em New
- 2- Clicar Add or upload data
- 3- Na guia de flies carregar ambos os arquivos
- 4- Clicar em Create Table with UI

New

Workspace

Recents

Search

Catalog

Workflows

Compute

Machine Learning

Experiments

Add data >

Create New Table

Data source

Upload File S3

DBFS Target Directory ?

/FileStore/tables/ (optional)

Select

Files uploaded to DBFS are accessible by everyone who has access to this workspace. [Learn more](#)

Files ?

athlete_events.csv

35.9 MB

Remove file

noc_regions.csv

3.8 KB

Remove file

✓File uploaded to /FileStore/tables/noc_regions-8.csv

✓File uploaded to /FileStore/tables/athlete_events_1-7.csv


Create Table with UI

Create Table in Notebook ?

5- Selecionar o Cluster – Engenharia de Dados

Select a Cluster to Preview the Table

Choose a cluster with which you will read and preview the data.

Cluster 

Cluster – Engenharia de Dados



Preview Table

6- Clicar Preview Table

Obs.: Seleccionadas as opções *"First row is header"*, para definir a primeira linha como cabeçalho, e *"Infer schema"*, para que os tipos das colunas sejam definidos automaticamente.

New

Workspace

Recents

Search

Catalog

Workflows

Compute

Machine Learning

Experiments

Select a Cluster to Preview the Table

Choose a cluster with which you will read and preview the data.

ClusterCluster – Engenharia de Dados

Preview Table

Specify Table Attributes

Specify the Table Name, Database and Schema to add this to the data UI for other users to access

Table Nameathlete_events_1_7_csv

Create in Databasedefault

File TypeCSV

Column Delimiter,

☒ First row is header

☒ Infer schema

☐ Multi-line

Create Table

Create Table in

Table Preview

ID	Name	Sex	Age	Height	Weight
INT	STRING	STRING	STRING	STRING	STRING
1	A Dijiang	M	24	180	80
2	A Lamusi	M	23	170	60
3	Gunnar Nielsen Aaby	M	24	NA	NA
4	Edgar Lindenaau Aabye	M	34	NA	NA
5	Christine Jacoba Aaftink	F	21	185	82
5	Christine Jacoba Aaftink	F	21	185	82

New

Workspace

Recents

Search

Catalog

Workflows

Compute

Machine Learning

Experiments

Select a Cluster to Preview the Table

Choose a cluster with which you will read and preview the data.

ClusterCluster – Engenharia de Dados

Preview Table

Specify Table Attributes

Specify the Table Name, Database and Schema to add this to the data UI for other users to access

Table Namenoc_regions_9_csv

Create in Databasedefault

File TypeCSV

Column Delimiter,

☒ First row is header

☒ Infer schema

☐ Multi-line

Create Table

Create Table in Notebook

Table Preview

NOC	region	notes
STRING	STRING	STRING
AFG	Afghanistan	null
AHO	Curacao	Netherlands Antilles
ALB	Albania	null
ALG	Algeria	null
AND	Andorra	null
ANG	Angola	null
ANT	Antigua	Antigua and Barbuda

B. Tabela noc_regions-9

noc_regions-9 Python

File Edit View Run Help Last edit was 1 hour ago

Run all Cluster - Engenharia d... Share

```
# File location and type
file_location = "/FileStore/tables/noc_regions-9.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
    .option("inferSchema", infer_schema) \
    .option("header", first_row_is_header) \
    .option("sep", delimiter) \
    .load(file_location)

display(df)
```

▶ (3) Spark Jobs

▶ df: pyspark.sql.dataframe.DataFrame = [NOC: string, region: string ... 1 more field]

Table

	NOC	region	notes
1	AFG	Afghanistan	null
2	AHO	Curacao	Netherlands Antilles
3	ALB	Albania	null
4	ALG	Algeria	null
5	AND	Andorra	null

ANÁLISES

Ao verificar a quantidade de valores nulos na coluna *Weight* (Peso), percebeu-se que muitos deles estão representados como "NA" em vez de nulos. Por isso, foi necessário ajustar o código para contabilizar tanto os valores nulos quanto aqueles marcados como "NA".

Code text

```
from pyspark.sql.functions import col

# Contar a quantidade de valores nulos na coluna "Weight"
qtd_nulos_weight = df.filter(col("Weight").isNull()).count()

print(f"Quantidade de valores nulos na coluna 'Weight': {qtd_nulos_weight}")

#Ao observar que os dados não estão nulos e estão como NA

# Conta os valores nulos ou "NA" na coluna "Weight"
qtd_nulos_na = df.filter((col("Weight").isNull()) | (col("Weight") == "NA")).count()

# Exibe o resultado
print(f"Quantidade de valores nulos ou 'NA' na coluna 'Weight': {qtd_nulos_na}")
```

▶ (4) Spark Jobs

Quantidade de valores nulos na coluna 'Weight': 0
Quantidade de valores nulos ou 'NA' na coluna 'Weight': 62743

Criada uma visualização temporária e verificando a quantidade de registros nulos ou iguais a NA.

10:42 AM (<1s)

3

#criando uma visualização temporária

df.createOrReplaceTempView("athlete_events_view")

Just now (2s)

4

%sql

SELECT
-- Conta a quantidade de registros com o valor NULL na coluna 'Age'
COUNT(CASE WHEN Age IS NULL OR Age = 'NA' THEN 1 END) AS registros_nulos_ou_NA,

-- Conta o total de registros na tabela 'athlete_events_5_csv'
COUNT(*) AS total_registros

-- Define a tabela em que a consulta será realizada
FROM athlete_events_view;

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [registros_nulos_ou_NA: long, total_registros: long]

Table

+

registros_nulos_ou_NA

total_registros

1

9441

271116

Vamos verificar em quais anos os dados de Peso (Weight) e Idade (Age) estão faltando;

4 minutes ago (4s)

5

SQL

%sql

SELECT DISTINCT Year,
COUNT(CASE WHEN Age IS NULL OR Age = 'NA' THEN 1 END) AS registros_nulos_Age, -- Contar os registros com valores nulos ou "NA" na coluna Age
COUNT(CASE WHEN Weight IS NULL OR Weight = 'NA' THEN 1 END) AS registros_nulos_Weight -- Contar os registros com valores nulos ou "NA" na
coluna Weight
FROM athlete_events_view -- Na tabela temporária criada para consultar os dados
WHERE Age IS NULL OR Age = 'NA' -- Filtrar as linhas onde a coluna Age é nula ou contém "NA"
OR Weight IS NULL OR Weight = 'NA' -- Filtrar as linhas onde a coluna Weight é nula ou contém "NA"
GROUP BY Year -- Agrupar os resultados por ano, para contar os nulos por ano
ORDER BY Year; -- Ordenar os resultados em ordem crescente de ano

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Year: string, registros_nulos_Age: long ... 1 more field]

Table

+

Year

registros_nulos_Age

registros_nulos_Weight

56

1992 Summer

0

3

57

1994

2

188

58

1994 Winter

0

1

59

1996

8

1809

60

1996 Summer

0

12

61

1998

2

86

62

2000

1

126

63

2002

0

47

64

2004

0

37

65

2006

0

16

Como era de se esperar os dados com o maior número de informações faltantes, são dados antigos, por isso para essa análise será considerado os dados do ano de 2000 até o ano de 2016.

Just now (2s) 6 SQL

```
%sql
SELECT
  MIN(CAST(Year AS INT)) AS Ano_Minimo, -- Obtém o menor ano da base, convertendo a coluna Year para INT
  MAX(CAST(Year AS INT)) AS Ano_Maximo -- Obtém o maior ano da base, convertendo a coluna Year para INT
FROM athlete_events_view; -- Nome da tabela ou view usada na consulta
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Ano_Minimo: integer, Ano_Maximo: integer]

Table

	Ano_Minimo	Ano_Maximo
1	1896	2016

1 row | 1.95s runtime Refreshed now

Para isso, foi criada uma tabela temporária com os dados de 2010 a 2016.

athlete_events_1-7.csv Python

File Edit View Run Help Last edit was now

Run all Cluster - Engenharia d... Share Publish

Just now (<1s) 7

```
%sql
-- Criando uma nova tabela temporária contendo apenas os anos de interesse
CREATE OR REPLACE TEMP VIEW athlete_events_filtered AS
SELECT *
FROM athlete_events_view -- Usando a tabela temporária original
WHERE CAST(Year AS INT) BETWEEN 2010 AND 2016; -- Filtrando os anos desejados
```

_sqldf: pyspark.sql.dataframe.DataFrame

OK

This result is stored as _sqldf and can be used in other Python cells.

Just now (1s) 8 SQL

```
%sql
-- Exibir os dados filtrados
SELECT * FROM athlete_events_filtered
LIMIT 100; -- Mostrar apenas as 100 primeiras linhas
```

(1) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [ID: integer, Name: string ... 13 more fields]

Table

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games
1	2	A Lamusi	M	23	170	60	China	CHN	2012 Su
2	16	Juhamatti Tapio Aaltonen	M	28	184	85	Finland	FIN	2014 Wi
3	22	Andreea Aanei	F	22	170	125	Romania	ROU	2016 Su

Criando uma tabela no catalog do databrick para que eu possa consultar em outro notebook, onde está sendo desenvolvido o código principal;

CSV Rows | 1.20s runtime

Just now (1s) 2

```
#criando uma visualização temporária
df.createOrReplaceTempView("noc_regions9")

# Salvar como arquivo Parquet
df.write.parquet("/dbfs/tmp/noc_regions9.parquet")
```

(1) Spark Jobs

Lendo o arquivo no notebook principal

```
Just now (1s) 9

# Ler o arquivo Parquet
df_noc = spark.read.parquet("/dbfs/tmp/noc_regions9.parquet")
df_noc.createOrReplaceTempView("noc_regions9")

(1) Spark Jobs

df_noc: pyspark.sql.dataframe.DataFrame = [NOC: string, region: string ... 1 more field]
```

Junção dos arquivos noc_regions9 e athlete_events_filtered

athlete_events_1-7.csv Python 02:02 PM (4s) 10 SQL

File Edit View Run Help Last edit was 10 minutes ago

Run all Cluster - Engenharia de... Share Publish

```
%sql
-- Utilizando SQL para fazer a junção entre as tabelas
SELECT a.*, b.* -- Selecionar todas as colunas das tabelas a e b (athlete_events_filtered e noc_regions9)
FROM athlete_events_filtered a
JOIN noc_regions9 b
ON a.NOC = b.NOC; -- Considerar a tabela NOC
```

(3) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [ID: integer, Name: string ... 16 more fields]

ID	Name	Sex	Age	Height	Weight	Team	NOC	Games
2	A Lamusi	M	23	170	60	China	CHN	2012 Su
16	Juhamatti Tapio Aaltonen	M	28	184	85	Finland	FIN	2014 Wi
22	Andreea Aanei	F	22	170	125	Romania	ROU	2016 Su
34	Jamale (Djamel-) Aarrass (Ahrass-)	M	30	187	76	France	FRA	2012 Su
48	Abdelhak Aatakni	M	24	NA	64	Morocco	MAR	2012 Su
51	Nistor Abad Sanjun	M	23	167	64	Spain	ESP	2016 Su
51	Nistor Abad Sanjun	M	23	167	64	Spain	ESP	2016 Su
51	Nistor Abad Sanjun	M	23	167	64	Spain	ESP	2016 Su
51	Nistor Abad Sanjun	M	23	167	64	Spain	ESP	2016 Su
51	Nistor Abad Sanjun	M	23	167	64	Spain	ESP	2016 Su
51	Nistor Abad Sanjun	M	23	167	64	Spain	ESP	2016 Su
55	Antonio Abadia Beci	M	26	170	65	Spain	ESP	2016 Su
62	Giovanni Abagnale	M	21	198	90	Italy	ITA	2016 Su
65	Patimat Abakarova	F	21	165	49	Azerbaijan	AZE	2016 Su

Verificação dos atletas mais pesados nessa temporalidade 2000-2016

```
%sql
SELECT Name, Weight, Sport, Year, NOC
FROM athlete_events_filtered
WHERE Weight IS NOT NULL -- Remove valores nulos
AND Weight <> 'NA' -- Remove registros com "NA"
ORDER BY CAST(Weight AS FLOAT) DESC -- Ordena pelo peso do maior para o menor
LIMIT 10; -- Retorna apenas os 10 primeiros registros
```

► (1) Spark Jobs

► _sqldf: pyspark.sql.dataframe.DataFrame = [Name: string, Weight: string ... 3 more fields]

Table ▼ +

	A _C Name	A _C Weight	A _C Sport	A _C Year	A _C NOC
1	Janusz Wojnarowicz	170	Judo	2012	POL
2	Illie Daniel Natea	170	Judo	2016	ROU
3	Ion Emilianov	165	Athletics	2016	MDA
4	Carl Andrew Myerscou...	160	Athletics	2012	GBR
5	Behdad Salimi Kordasia...	160	Weightlifting	2012	IRI
6	Pter Nagy	160	Weightlifting	2012	HUN
7	Rafael Carlos da Silva	160	Judo	2012	BRA
8	Pter Nagy	160	Weightlifting	2016	HUN
9	Lasha Talakhadze	160	Weightlifting	2016	GEO
10	Rafael Carlos da Silva	160	Judo	2016	BRA

↓ 10 rows | 3.46s runtime

ⓘ This result is stored as _sqldf and can be used in other Python cells.

Relação Idade x Quantidade de Medalhas

athlete_events_1-7.csv Python

File Edit View Run Help Last edit was now

► Run all Cluster - Engenharia de... Share Publish

3 minutes ago (3s) 12

```
%sql
-- Seleciona a idade do atleta, o tipo de medalha e a contagem de medalhas
SELECT Age, Medal, COUNT(*) AS Total_Medalhas
FROM athlete_events_filtered
WHERE Age IS NOT NULL -- Filtrar registros onde a idade não é nula
AND Age <> 'NA' -- Remover registros onde a idade contém "NA"
AND Medal IN ('Gold', 'Silver', 'Bronze') -- Considerar apenas medalhas válidas (ouro, prata e bronze)
GROUP BY Age, Medal -- Agrupar os resultados por idade e tipo de medalha
ORDER BY Total_Medalhas DESC; -- Ordenar os resultados pela idade de forma crescente
```

► (2) Spark Jobs

► _sqldf: pyspark.sql.dataframe.DataFrame = [Age: string, Medal: string ... 1 more field]

Table ▼ +

	A _C Age	A _C Medal	T ₃ Total_Medalhas
1	25	Silver	144
2	24	Gold	135
3	27	Gold	125
4	25	Bronze	125
5	24	Silver	121
6	26	Gold	117
7	26	Bronze	117
8	23	Gold	116
9	27	Silver	115
10	27	Bronze	115
11	23	Bronze	114

Quantidade de atletas por idade

athlete_events_1-7.csv Python

File Edit View Run Help Last edit was now

Run all Cluster - Engenharia de... Share Publish

Just now (4s) 13

```
%sql
-- Seleciona a idade e a contagem de atletas distintos
SELECT Age, COUNT(DISTINCT Name) AS Total_Atletas
FROM athlete_events_filtered
WHERE Age IS NOT NULL -- Filtrar registros onde a idade não é nula
AND Age <> 'NA' -- Remover registros onde a idade contém "NA"
GROUP BY Age -- Agrupar os resultados por idade
ORDER BY Total_Atletas DESC; -- Ordenar pela idade em ordem crescente
```

(3) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Age: string, Total_Atletas: long]

Table +

	^A _C Age	¹ ₃ Total_Atletas
1	25	2207
2	24	2185
3	23	2159
4	26	2041
5	27	2019
6	22	1986
7	28	1740
8	21	1604
9	29	1566
10	30	1284
11	20	1192

Peso médio ao longo dos anos e por esporte

Just now (4s) 14

```
%sql
-- Analisar o peso médio dos atletas por ano e por esporte
SELECT
  Year, -- Ano da edição olímpica
  Sport, -- Modalidade esportiva
  COUNT(*) AS Total_Atletas, -- Total de atletas registrados naquela combinação de ano e esporte
  AVG(CAST(Weight AS FLOAT)) AS Peso_Medio -- Peso médio dos atletas (convertido para float)
FROM athlete_events_filtered
WHERE Weight IS NOT NULL -- Excluir valores nulos
AND Weight <> 'NA' -- Excluir registros com "NA"
GROUP BY Year, Sport -- Agrupar por ano e esporte
ORDER BY Year, Peso_Medio DESC; -- Ordenar por ano e maior peso médio
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Year: string, Sport: string ... 2 more fields]

Table +

	^A _C Year	^A _C Sport	¹ ₃ Total_Atletas	¹ ₂ Peso_Medio
1	2010	Bobsleigh	199	92.92713567839196
2	2010	Ice Hockey	417	81.00599520383693
3	2010	Luge	107	79.51401869158879
4	2010	Alpine Skiing	681	75.33480176211454
5	2010	Skeleton	47	72.8936170212766
6	2010	Curling	89	71.79775280898876

Relação entre altura e esporte

Just now (3s) 15

```
%sql
-- Verificar a altura média dos atletas por esporte
SELECT
  Sport,
  COUNT(*) AS Total_Atletas,
  AVG(CAST(Height AS FLOAT)) AS Altura_Media
FROM athlete_events_filtered
WHERE Height IS NOT NULL
  AND Height <> 'NA'
GROUP BY Sport
ORDER BY Altura_Media DESC;
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Sport: string, Total_Atletas: long ... 1 more field]

Table +

	A _C Sport	1 ² ₃ Total_Atletas	1.2 Altura_Media
1	Basketball	561	192.06595365418895
2	Volleyball	570	190.1280701754386
3	Beach Volleyball	192	187.15625
4	Water Polo	515	185.68543689320387
5	Rowing	1094	185.28519195612432
6	Handball	698	183.76647564469914
7	Bobsleigh	421	182.82422802850357
8	Swimming	3035	180.54530477759474
9	Tennis	569	180.1476274165202

Altura média dos medalhistas de ouro por esporte

Just now (3s) 16

```
%sql
-- Calcular a altura média apenas dos atletas que ganharam medalha de ouro
SELECT
  Sport,
  COUNT(*) AS Medalhistas_Ouro,
  AVG(CAST(Height AS FLOAT)) AS Altura_Media_Ouro
FROM athlete_events_filtered
WHERE Medal = 'Gold'
  AND Height IS NOT NULL
  AND Height <> 'NA'
GROUP BY Sport
ORDER BY Altura_Media_Ouro DESC;
```

(2) Spark Jobs

_sqldf: pyspark.sql.dataframe.DataFrame = [Sport: string, Medalhistas_Ouro: long ... 1 more field]

Table +

	A _C Sport	1 ² ₃ Medalhistas_Ouro	1.2 Altura_Media_Ouro
1	Basketball	44	192.931818181818182
2	Volleyball	48	192.895833333333334
3	Water Polo	51	188.3921568627451
4	Beach Volleyball	8	188.25
5	Rowing	96	186.572916666666666
6	Bobsleigh	16	185.25
7	Swimming	129	185.05426356589146

Relação entre esporte e quantidade de edições dos Jogos

```

%sql
-- Verificar em quantas edições dos Jogos cada esporte apareceu
SELECT
  Sport,                                -- Modalidade esportiva
  COUNT(DISTINCT Year) AS Total_Edicoes -- Número de edições olímpicas em que o esporte esteve presente
FROM athlete_events_filtered
GROUP BY Sport                          -- Agrupamos por esporte
ORDER BY Total_Edicoes DESC;           -- Ordenamos do que mais apareceu para o que menos apareceu

```

(3) Spark Jobs

_sqlIdf: pyspark.sql.dataframe.DataFrame = [Sport: string, Total_Edicoes: long]

	Sport	Total_Edicoes
1	Gymnastics	2
2	Tennis	2
3	Boxing	2
4	Ice Hockey	2
5	Rowing	2
6	Judo	2
7	Sailing	2
8	Swimming	2
9	Alpine Skiing	2
10	Basketball	2

GRÁFICOS

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Configurações visuais
sns.set(style="whitegrid")
plt.figure(figsize=(18, 6))

# Carregar os dados tratados como Pandas
df_athletes = spark.sql("""
    SELECT *
    FROM athlete_events_filtered
    WHERE Weight IS NOT NULL AND Weight != 'NA'
       AND Age IS NOT NULL AND Age != 'NA'
       AND Height IS NOT NULL AND Height != 'NA'
""").toPandas()

# Conversões
df_athletes["Weight"] = df_athletes["Weight"].astype(float)
df_athletes["Age"] = df_athletes["Age"].astype(int)
df_athletes["Height"] = df_athletes["Height"].astype(float)

### Análise 1: Atletas mais pesados
print(" Análise 1: Top 10 Atletas Mais Pesados")
top_pesados = df_athletes.sort_values(by="Weight", ascending=False).head(10)
plt.figure(figsize=(10,6))
sns.barplot(x="Weight", y="Name", data=top_pesados, hue="Sport", dodge=False)
plt.title("Top 10 Atletas Mais Pesados")
```

```

plt.figure(figsize=(10,6))
sns.barplot(x="Weight", y="Name", data=top_pesados, hue="Sport", dodge=False)
plt.title("Top 10 Atletas Mais Pesados")
plt.xlabel("Peso (kg)")
plt.ylabel("Nome")
plt.legend(title="Esporte")
plt.show()

### Análise 2: Idade x Medalhas
print(" Análise 2: Relação entre Idade e Quantidade de Medalhas")
medalhas = df_athletes[df_athletes["Medal"].isin(["Gold", "Silver", "Bronze"])]
idade_medalhas = medalhas.groupby("Age")["Medal"].count().reset_index(name="Total_Medalhas")
plt.figure(figsize=(10,5))
sns.lineplot(x="Age", y="Total_Medalhas", data=idade_medalhas, marker="o")
plt.title("Quantidade de Medalhas por Idade")
plt.xlabel("Idade")
plt.ylabel("Total de Medalhas")
plt.show()

### Análise 3: Distribuição da Idade dos Atletas
print(" Análise 3: Distribuição da Idade dos Atletas")
plt.figure(figsize=(10,5))
sns.histplot(df_athletes["Age"], bins=30, kde=True)
plt.title("Distribuição da Idade dos Atletas")
plt.xlabel("Idade")
plt.ylabel("Quantidade de Atletas")
plt.show()

### Análise 4: Altura x Tipo de Esporte
print(" Análise 4: Relação entre Altura e Esporte")

```

```

### Análise 4: Altura x Tipo de Esporte
print(" Análise 4: Relação entre Altura e Esporte")
top_sports = df_athletes["Sport"].value_counts().head(10).index
altura_esportes = df_athletes[df_athletes["Sport"].isin(top_sports)]
plt.figure(figsize=(12,6))
sns.boxplot(data=altura_esportes, x="Sport", y="Height")
plt.xticks(rotation=45)
plt.title("Altura por Esporte (Top 10 Modalidades)")
plt.show()

### Análise 5: Peso Médio por Ano e Esporte com Maior Peso Médio
print(" Análise 5: Peso Médio por Ano")
peso_ano = df_athletes.groupby("Year")["Weight"].mean().reset_index()
plt.figure(figsize=(10,5))
sns.lineplot(x="Year", y="Weight", data=peso_ano, marker="o")
plt.title("Variação do Peso Médio dos Atletas ao Longo dos Anos")
plt.xlabel("Ano")
plt.ylabel("Peso Médio (kg)")
plt.show()

print(" Esportes com Maior Peso Médio")
peso_sport = df_athletes.groupby("Sport")["Weight"].mean().sort_values(ascending=False).head(10)
peso_sport.plot(kind='barh', figsize=(10,5), title="Top 10 Esportes com Maior Peso Médio")
plt.xlabel("Peso Médio (kg)")
plt.gca().invert_yaxis()
plt.show()

### Análise 6: Altura Média dos Medalhistas de Ouro por Esporte
print(" Análise 6: Altura Média dos Medalhistas de Ouro")
ouro = df_athletes[df_athletes["Medal"] == "Gold"]
altura_media_ouro = ouro.groupby("Sport")["Height"].mean().sort_values(ascending=False).head(10)

```

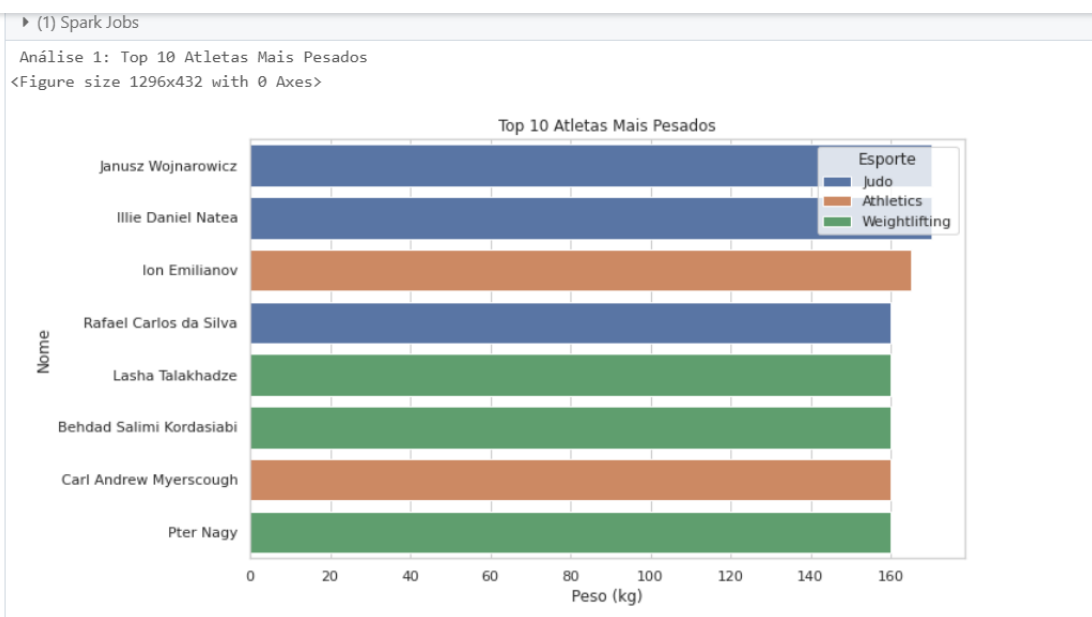
```
Just now (6s) 18

### Análise 6: Altura Média dos Medalhistas de Ouro por Esporte
print(" Análise 6: Altura Média dos Medalhistas de Ouro")
ouro = df_athletes[df_athletes["Medal"] == "Gold"]
altura_media_ouro = ouro.groupby("Sport")["Height"].mean().sort_values(ascending=False).head(10)
altura_media_ouro.plot(kind="bar", figsize=(10,5), title="Altura Média dos Medalhistas de Ouro (Top 10)")
plt.ylabel("Altura Média (cm)")
plt.xticks(rotation=45)
plt.show()

### Análise 7: Esportes com mais edições
print(" Análise 7: Esporte x Quantidade de Jogos (edições)")
esporte_jogos = df_athletes.groupby("Sport")["Year"].nunique().sort_values(ascending=False).head(10)
esporte_jogos.plot(kind="bar", figsize=(10,5), title="Top 10 Esportes com mais Edições Olímpicas")
plt.ylabel("Número de Edições")
plt.xticks(rotation=45)
plt.show()

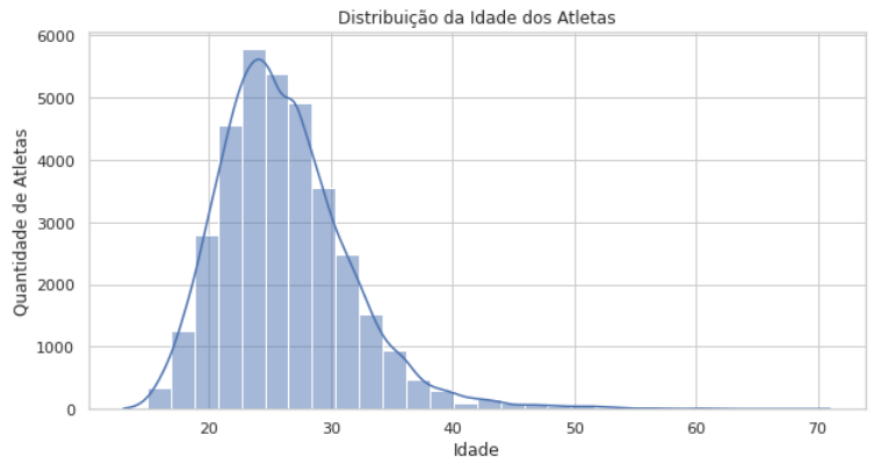
### Análise 8: Atletas Brasileiros com Mais Participações
print(" Análise 8: Atletas Brasileiros com Mais Jogos")
brasil = df_athletes[df_athletes["NOC"] == "BRA"]
participacoes = brasil.groupby("Name")["Year"].nunique().sort_values(ascending=False).head(10)
participacoes.plot(kind="bar", figsize=(10,5), title="Top 10 Atletas Brasileiros com Mais Participações")
plt.ylabel("Número de Edições")
plt.xticks(rotation=45)
plt.show()

▶ (1) Spark Jobs
```

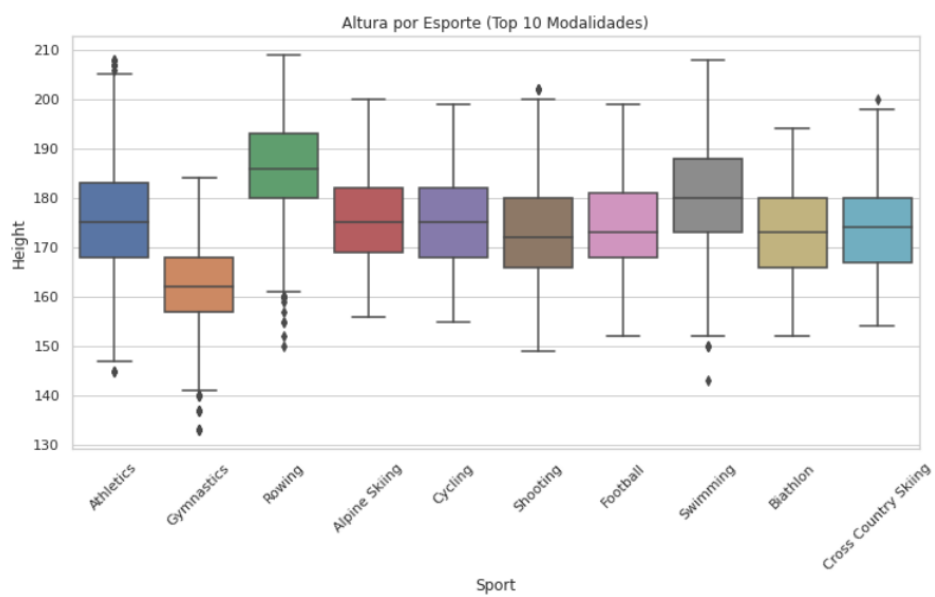


Análise 2: Relação entre Idade e Quantidade de Medalhas





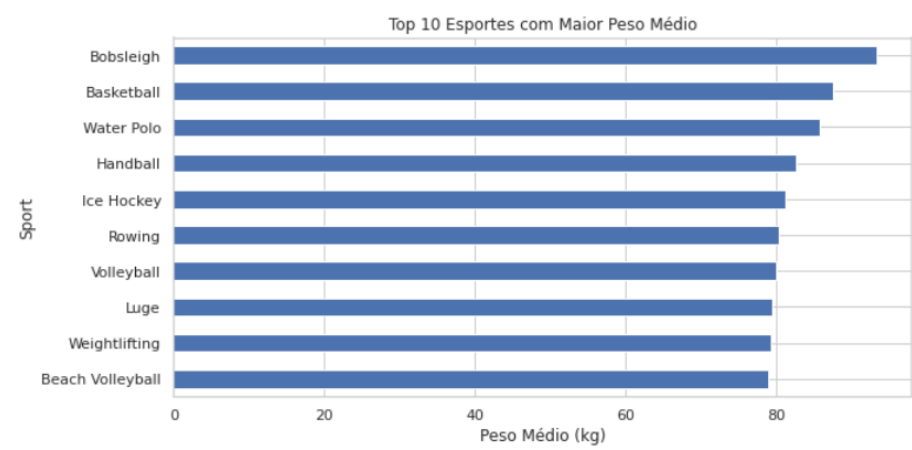
Análise 4: Relação entre Altura e Esporte



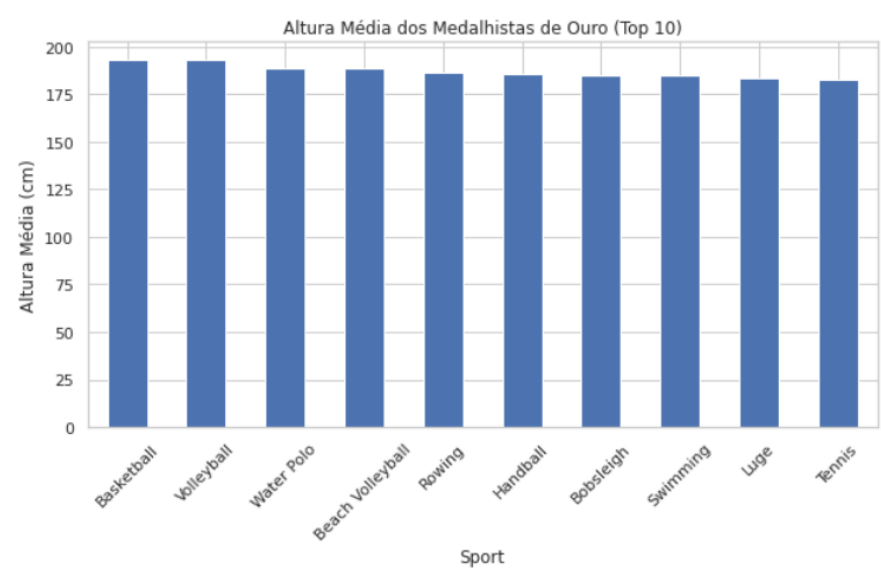
Análise 5: Peso Médio por Ano



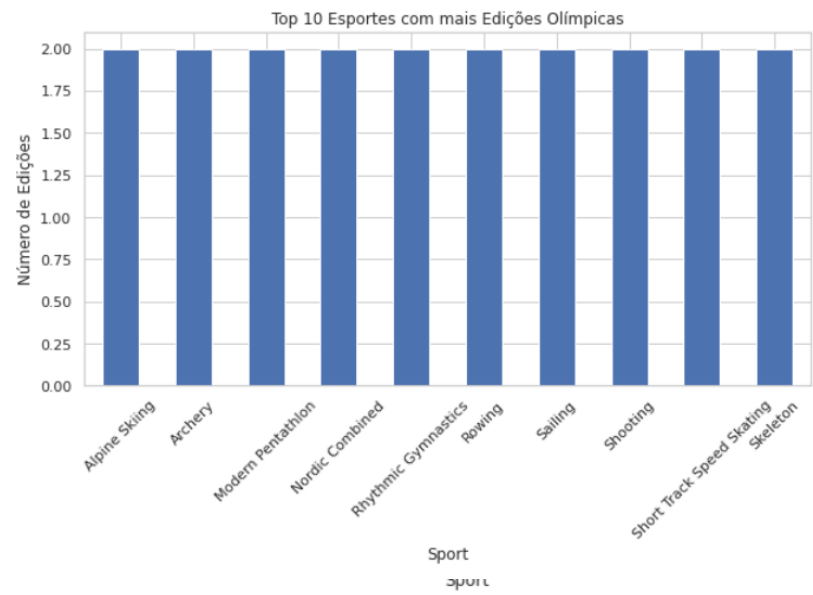
Esportes com Maior Peso Médio



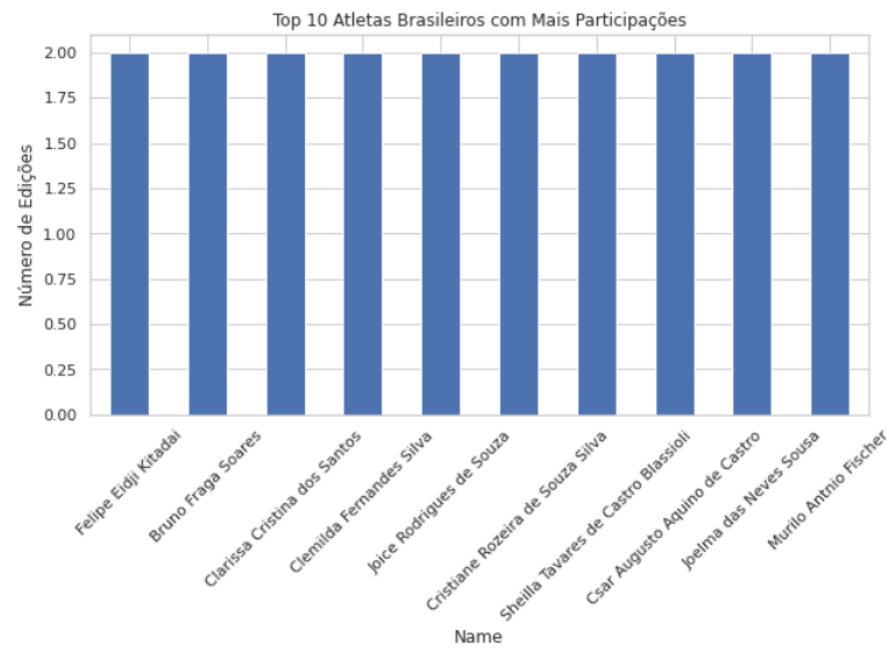
Análise 6: Altura Média dos Medalhistas de Ouro



Análise 7: Esporte x Quantidade de Jogos (edições)



Análise 8: Atletas Brasileiros com Mais Jogos



CONSIDERAÇÕES FINAIS E AUTOAVALIAÇÃO

Este trabalho permitiu uma visão abrangente sobre os dados históricos dos Jogos Olímpicos, possibilitando análises relevantes sobre o perfil dos atletas, desempenho por país e curiosidades envolvendo idade, peso, altura e medalhas. Foi utilizado ferramentas como PySpark e SQL no ambiente Databricks para explorar e visualizar os dados de forma eficiente.

Durante o desenvolvimento, foi possível aplicar conceitos de tratamento de dados, criação de tabelas temporárias, filtros, joins e visualizações. As perguntas propostas foram, em sua maioria, respondidas com base em análises reais e dados concretos.

Do ponto de vista pessoal, avalio que o trabalho foi bem executado. Consegui organizar bem as etapas, interpretar os dados e aplicar os conhecimentos adquiridos ao longo do curso. Ainda há espaço para aprofundar algumas análises e melhorar a interpretação de resultados, mas, de forma geral, o objetivo do trabalho foi alcançado com sucesso.