

Pipeline de dados da Felicidade Mundial

Github: <https://github.com/fabsy381/mvp>

20250413



Escopo - Pipeline de Dados da Felicidade Mundial

O presente projeto tem como objetivo desenvolver um pipeline de dados, com base no conjunto de dados do **Relatório Mundial da Felicidade (World Happiness Report)** entre os anos de **2015 a 2023**. O relatório é uma publicação anual de destaque que oferece rankings de felicidade mundial e análises sobre o bem-estar global. Publicado desde 2012, ele se tornou uma referência importante para entender e medir a felicidade e o bem-estar em diferentes países. O foco principal do pipeline é organizar, tratar, modelar e analisar as informações relativas à **felicidade mundial**, visando compreender os fatores que influenciam esse índice e possibilitar **análises exploratórias** para gerar **insights relevantes**.

Objetivo

As análises do estudo se concentram no “ranking da felicidade” (Happiness Rank), que representa a pontuação do índice de felicidade para cada país. O conjunto de dados é verificado com base nos fatores, como PIB per capita, suporte social, liberdade para fazer escolhas de vida, expectativa de vida saudável ao nascer, medida de inflação, generosidade e percepções de corrupção. Compreender os fatores que impactam o índice de felicidade dos países ao longo do tempo e responder perguntas-chave de negócio a partir de dados públicos.

- Avaliar a evolução da felicidade global de forma estruturada
- Identificar padrões entre países mais e menos felizes
- Relacionar indicadores socioeconômicos com níveis de felicidade

Perguntas de Negócio

A partir da modelagem dimensional e da camada Gold do pipeline, este projeto busca responder às seguintes perguntas:

1. Qual a evolução da felicidade média no mundo por ano?
2. Comparativo de felicidade média por continente em 2023?
3. Top 10 países mais felizes em 2023?
4. Fatores mais presentes entre os países mais felizes?
5. Top 10 países menos felizes em 2023
6. Quais fatores estão mais presentes entre os países menos felizes?
7. Quais são os 5 países menos felizes em cada ano?
8. Existe relação entre PIB per capita e felicidade?
9. Como a expectativa de vida afeta a felicidade?
10. Quais países se destacam por generosidade ou suporte social?
11. Correlação entre felicidade e GDP em 2023
12. Variação da felicidade no Brasil por ano?
13. Quais fatores estão mais presentes na felicidade do Brasil?
14. Comparativo: Brasil vs América Latina e Caribe



Conjunto de Dados

Felicidade e bem-estar são indicadores essenciais do progresso social, frequentemente influenciados por condições econômicas como PIB e inflação. Este conjunto de dados combina dados do Índice Mundial de Felicidade (WHI) e métricas de inflação para explorar a relação entre estabilidade econômica e níveis de felicidade de 2015 a 2023. O conjunto de dados é um ranking de classificação de países por felicidade, e pontuação dos fatores que contribuem para os níveis de felicidade como: PIB per capita, suporte social, expectativa de vida saudável ao nascer, liberdade para fazer escolhas de vida, generosidade, percepções de corrupção e várias medidas de inflação.

Fonte dos Dados

O conjunto de dados utilizado neste projeto foi obtido na plataforma Kaggle, no seguinte repositório público:

 [World Happiness Report 2015–2023 – Kaggle Dataset](#)




A base de dados está organizada em dois arquivos principais:

1. `WHRFinal.csv`: Contém dados de felicidade de diversos países entre os anos de 2015 a 2023, com os indicadores:
 - PIB per capita (`gdp`)
 - Suporte social (`social_support`)
 - Expectativa de vida (`life_expectancy`)
 - Liberdade (`freedom`)
 - Generosidade (`generosity`)
 - Percepção de corrupção (`corruption`)
 - Índice e ranking de felicidade
2. `WHRContinent.csv`: Arquivo auxiliar com a correspondência entre regiões e continentes, utilizado para enriquecer a análise geográfica.

Plataforma de Armazenamento e Processamento

A plataforma utilizada foi o **Databricks Community Edition**, que fornece um ambiente gratuito baseado na nuvem para análise de dados com Spark, Python e SQL.

Etapas realizadas:

1. **Upload dos arquivos CSV** diretamente no repositório `/FileStore/tables/` do ambiente Databricks.
2. **Leitura dos dados** com `PySpark`, incluindo configuração do separador (`;`), detecção de tipos e remoção de caracteres inválidos.
3. **Persistência** dos dados em diferentes camadas do pipeline:
 -  **Bronze**: Dados brutos, conforme extraídos.
 -  **Silver**: Dados tratados, limpos e tipados.
 -  **Gold**: Modelagem dimensional para análise (tabelas fato e dimensão).
4. **Armazenamento final** no **Databricks File System (DBFS)** em formato Delta Lake, permitindo reutilização e performance em consultas SQL.

Importação de Bibliotecas

- **Pandas** e **NumPy**: manipulação de dados e arrays
- **PySpark**: para processamento distribuído
- **Funções do PySpark**: para transformações
- **Matplotlib** e **Seaborn**: criar visualizações mais detalhadas

4: Importação bibliotecas

```
# Bibliotecas
import pandas as pd
import numpy as np

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when, isnan, count, lit, avg

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

Etapa 1 — Camada Bronze

Objetivo:

Carregar os datasets CSV da Felicidade Mundial e mapeamento de continentes para a camada Bronze do Data Lake.

Arquivos

- **WHRFinal.csv**: Dados de felicidade mundial (2015–2023)
- **WHRContinent.csv**: De-para de região vs. continente

Ações

- Leitura dos arquivos CSV com PySpark
- Inferência de schema e inclusão de headers
- Salvamento no formato Delta
- Registro como tabelas no metastore Databricks

6: Leitura datasets

```
# Leitura do dataset principal
df_happiness = spark.read.format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .load("/FileStore/tables/dataset/WHRFinal.csv")

# Leitura do dataset de continentes
df_continent = spark.read.format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .load("/FileStore/tables/dataset/WHRContinent.csv")

# Visualização
df_happiness.show(5)
df_continent.show(5)
```

df_continent: pyspark.sql.dataframe.DataFrame = [Region;Continent: string]

df_happiness: pyspark.sql.dataframe.DataFrame = [Country;Year;Region;Happiness Rank;Happiness Score;GDP;Social Support;Life Expectancy;Freedom;Generosity;Corruption: string]

```
|Country;Year;Region;Happiness Rank;Happiness Score;GDP;Social Support;Life Expectancy;Freedom;Generosity;Corruption|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                |                |                |                |                |                |                |                |                |                |
|                |                |                |                |                |                |                |                |                |
|                |                |                |                |                |                |                |                |                |
|                |                |                |                |                |                |                |                |                |
|                |                |                |                |                |                |                |                |                |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

only showing top 5 rows

```
+-----+-----+
|  Region;Continent|
+-----+-----+
| Southern Asia;Asia|
|Central and East...|
|Middle East and N...|
|Latin America and...|
|Australia and New...|
+-----+-----+
```

only showing top 5 rows

Normalização dos nomes de colunas

Os arquivos CSV possuem nomes de colunas com espaços e caracteres especiais, que não são compatíveis com a criação de tabelas Delta.Será realizado:

- remover espaços e caracteres especiais
- colocar os nomes em minúsculo com underscores (snake_case)
- salvar os dados na camada Bronze.

8: Formatação nome das colunas

```
# Função para limpar nomes de colunas
def clean_column_names(df):
    for col_name in df.columns:
        new_col = col_name.strip().lower().replace(" ", "_").replace("(", "").replace(")", "")
        df = df.withColumnRenamed(col_name, new_col)
    return df
```

9: Aplicar código

```
# Limpar colunas dos dois DataFrames
df_happiness_clean = clean_column_names(df_happiness)
df_continent_clean = clean_column_names(df_continent)

# Visualizar novos nomes
df_happiness_clean.printSchema()
df_continent_clean.printSchema()
```

```
▶ df_continent_clean: pyspark.sql.dataframe.DataFrame = [region;continent: string]
▶ df_happiness_clean: pyspark.sql.dataframe.DataFrame = [country;year;region;happiness_rank;happiness_score;gdp;social_support;life_expectancy;freedom;generosity;corruption: string]

root
|-- country;year;region;happiness_rank;happiness_score;gdp;social_support;life_expectancy;freedom;generosity;corruption: string (nullable = true)

root
|-- region;continent: string (nullable = true)
```

10: Salvando os dados como tabelas da camada Bronze

```
1 # Salvando dataset principal
2 df_happiness_clean.write.format("delta").mode("overwrite").save("/mnt/bronze/happiness")
3 spark.sql("DROP TABLE IF EXISTS bronze_happiness")
4 spark.sql("""
5     CREATE TABLE bronze_happiness
6     USING DELTA
7     LOCATION '/mnt/bronze/happiness'
8 """)
9
10 # Salvando dataset de continentes
11 df_continent_clean.write.format("delta").mode("overwrite").save("/mnt/bronze/continent")
12 spark.sql("DROP TABLE IF EXISTS bronze_continent")
13 spark.sql("""
14     CREATE TABLE bronze_continent
15     USING DELTA
16     LOCATION '/mnt/bronze/continent'
17 """)
18
```

AnalysisException: A schema mismatch detected when writing to the Delta table (Table ID: a9a1d4bb-00ea-44e3-85a8-141ce0e58a22).

To enable schema migration using DataStreamWriter or DataStreamWriter, please set:
'option("mergeSchema", "true")'....

Padronização de nomes de colunas

Foi identificado um erro ao tentar salvar os dados no formato Delta, devido à presença de caracteres inválidos nos nomes das colunas, será realizado:

- função que remove qualquer caractere não alfanumérico
- nomes convertidos para `snake_case`
- após a limpeza, os dados serão salvos na camada Bronze

13: Formatação nome das colunas

```
#remove tudo que não for letra, número ou underscore (_) dos nomes das colunas
import re
def sanitize_column_names(df):
    new_columns = []
    for col_name in df.columns:
        # Substitui espaços e caracteres especiais por underscore, remove acentos
        clean_name = re.sub(r"^\w+", "_", col_name.strip())
        clean_name = re.sub(r"_{+}", "_", clean_name) # Remove múltiplos underscores
        clean_name = clean_name.lower()
        new_columns.append(clean_name)

    for old_col, new_col in zip(df.columns, new_columns):
        df = df.withColumnRenamed(old_col, new_col)

    return df
```

14: Aplicar no Dataframe

```
# Aplicar padronização
df_happiness_clean = sanitize_column_names(df_happiness)
df_continent_clean = sanitize_column_names(df_continent)

# Verificar nomes
df_happiness_clean.printSchema()
df_continent_clean.printSchema()
```

```
▶ df_continent_clean: pyspark.sql.dataframe.DataFrame = [region_continent: string]
▶ df_happiness_clean: pyspark.sql.dataframe.DataFrame = [country_year_region_happiness_rank_happiness_score_gdp_social_support_life_expectancy_freedom_generosity_corruption: string]

root
|-- country_year_region_happiness_rank_happiness_score_gdp_social_support_life_expectancy_freedom_generosity_corruption: string (nullable = true)

root
|-- region_continent: string (nullable = true)
```


15: Salvar como Delta

```
# Dataset principal
df_happiness_clean.write.format("delta").mode("overwrite").save("/mnt/bronze/happiness")
spark.sql("DROP TABLE IF EXISTS bronze_happiness")
spark.sql("""
CREATE TABLE bronze_happiness
USING DELTA
LOCATION '/mnt/bronze/happiness'
""")

# Dataset de continentes
df_continent_clean.write.format("delta").mode("overwrite").save("/mnt/bronze/continent")
spark.sql("DROP TABLE IF EXISTS bronze_continent")
spark.sql("""
CREATE TABLE bronze_continent
USING DELTA
LOCATION '/mnt/bronze/continent'
""")
```

Out[165]: DataFrame[]

Validação dos dados da Camada Bronze com SQL

Após salvar os dados na camada Bronze, realizar consultas no Databricks para garantir que:

- os dados foram importados corretamente
- os tipos das colunas estão coerentes
- a quantidade de registros e países está dentro do esperado
- as tabelas podem ser usadas para análises posteriores

17: Verificar os dados da Bronze

```
%sql
SELECT * FROM bronze_happiness LIMIT 5;
```

▸  _sqldf: pyspark.sql.dataframe.DataFrame = [country_year_region_happiness_rank_happiness_score_gdp_social_support_life_expectancy_freedom_generosity_corruption: string]

Table




	^A _C country_year_region_happiness_rank_happiness_score_gdp_social_support_life_expectancy_freedom_generosity_corruption
1	Afghanistan;2023;Southern Asia;137;1
2	Afghanistan;2017;Southern Asia;141;3
3	Afghanistan;2018;Southern Asia;145;3
4	Afghanistan;2022;Southern Asia;146;2
5	Afghanistan;2021;Southern Asia;149;2

5 rows

18: Verificar columnas e tipos

```
%sql
DESCRIBE TABLE bronze_happiness;
```

▸  _sqldf: pyspark.sql.dataframe.DataFrame = [col_name: string, data_type: string ... 1 more field]

Table



	^A _C col_name	^A _C data_type	^A _C comment
1	country_year_region_happiness_rank_happiness_score_gdp_social_support_life_expectancy_freedom_generosity_corrupti...	string	null

1 row

19: Verificar os dados da Bronze

%sql

```
SELECT * FROM bronze_continent LIMIT 5;
```

►  _sqldf: pyspark.sql.dataframe.DataFrame = [region_continent: string]

Table


	^A _C region_continent
1	Southern Asia;Asia
2	Central and Eastern Europe;Europe
3	Middle East and Northern Africa;Middle East and Northern Afri...
4	Latin America and Caribbean;Latin America and Caribbean
5	Australia and New Zealand;Oceania

5 rows

20: Verificar colunas e tipos

%sql

```
DESCRIBE TABLE bronze_continent;
```

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

Table

	^A _C col_name	^A _C data_type	^A _C comment
1	region_continent	string	null

1 row

Etapa 2 — Camada Silver

Objetivo

Transformar os dados brutos da camada Bronze em dados limpos, padronizados e enriquecidos com informações geográficas (continente).

Será realizado

- Remoção de registros nulos em colunas críticas
- Padronização dos nomes de país e região
- Conversão correta dos tipos de dados
- Inclusão da coluna de continente (join com bronze_continent)
- Salvamento da tabela tratada no formato Delta

Modelo dimensional (estrela)

Criar um modelo dimensional (estrela) para suportar análises exploratórias dos dados de felicidade mundial, considerando aspectos geográficos, temporais e indicadores sociais.

Estrutura do modelo dimensional


- **Fato:** fato_felicidade — Contém os indicadores por país e ano
- **Dimensões:**
 - dim_pais — Informações geográficas: país, região, continente
 - dim_tempo — Informações temporais: ano, possíveis agregações futuras

22: Código para construção da Silver

```

1 from pyspark.sql.functions import col, trim
2
3 # Carregar as tabelas da Bronze
4 df_bronze_happiness = spark.read.format("delta").load("/mnt/bronze/happiness")
5 df_bronze_continent = spark.read.format("delta").load("/mnt/bronze/continent")
6
7 # Limpeza e tratamento (Silver)
8 df_silver = (
9     df_bronze_happiness
10     .dropna(subset=["country", "year", "happiness_score"])
11     .withColumn("country", trim(col("country")))
12     .withColumn("region", trim(col("region")))
13     .withColumn("year", col("year").cast("int"))
14     .withColumn("happiness_score", col("happiness_score").cast("float"))
15     .withColumn("gdp", col("gdp").cast("float"))
16     .withColumn("social_support", col("social_support").cast("float"))
17     .withColumn("life_expectancy", col("life_expectancy").cast("float"))
18     .withColumn("freedom", col("freedom").cast("float"))
19     .withColumn("generosity", col("generosity").cast("float"))
20     .withColumn("corruption", col("corruption").cast("float"))
21 )
22
23 # Enriquecimento com continente
24 df_silver = df_silver.join(
25     df_bronze_continent,
26     on="region",
27     how="left"
28 )
29

```

 > **AnalysisException: [UNRESOLVED_COLUMN.WITH_SUGGESTION]** A column or function parameter with name `country` cannot be resolved. Did you mean one of the following? [`country_year_region_happiness_rank_happiness_score_gdp_social_support_life_expectancy_freedom_generosity_corruption`].

23: Releitura do csv

```

# Corrigir o erro (como se todo o cabeçalho do CSV tivesse sido lido como um único campo)
# Refazer a leitura da Bronze
df_bronze_happiness = spark.read.format("csv") \
    .option("header", "true") \
    .option("sep", ",") \
    .option("inferSchema", "true") \
    .load("/FileStore/tables/dataset/WHRFinal.csv")

df_bronze_continent = spark.read.format("csv") \
    .option("header", "true") \
    .option("sep", ",") \
    .option("inferSchema", "true") \
    .load("/FileStore/tables/dataset/WHRContinent.csv")

```

▸  df_bronze_continent: pyspark.sql.dataframe.DataFrame = [Region;Continent: string]

▸  df_bronze_happiness: pyspark.sql.dataframe.DataFrame = [Country;Year;Region;Happiness Rank;Happiness Score;GDP;Social Support;Life Expectancy;Freedom;Generosity;Corruption: string]

24: Verificar as colunas

```
#Verifique se as colunas estão no formato correto
```

```
df_bronze_happiness.printSchema()
```

```
df_bronze_happiness.show(5)
```

```
root
```

```
|-- Country;Year;Region;Happiness Rank;Happiness Score;GDP;Social Support;Life Expectancy;Freedom;Generosity;Corruption: string (nullable = true)
```

```
+-----+
|Country;Year;Region;Happiness Rank;Happiness Score;GDP;Social Support;Life Expectancy;Freedom;Generosity;Corruption|
+-----+
|                                                                                               Afghanistan;2023;...|
|                                                                                               Afghanistan;2017;...|
|                                                                                               Afghanistan;2018;...|
|                                                                                               Afghanistan;2022;...|
|                                                                                               Afghanistan;2021;...|
+-----+
```

```
only showing top 5 rows
```

25: Reaplicar a função de padronização

```
# Reaplicar a função de padronização nas colunas
```

```
df_bronze_happiness = sanitize_column_names(df_bronze_happiness)
```

```
df_bronze_continent = sanitize_column_names(df_bronze_continent)
```

```
▶ df_bronze_continent: pyspark.sql.dataframe.DataFrame = [region_continent: string]
```

```
▶ df_bronze_happiness: pyspark.sql.dataframe.DataFrame = [country_year_region_happiness_rank_happiness_score_gdp_social_support_life_expectancy_freedom_generosity_corruption: string]
```

26: Corrigir separador correto (;) na leitura do CSV

```
# Especificar o separador correto (;) na leitura do CSV
```

```
# Lendo corretamente com separador ";"
```

```
df_bronze_happiness = spark.read.format("csv") \
    .option("header", "true") \
    .option("sep", ";") \
    .option("inferSchema", "true") \
    .load("/FileStore/tables/dataset/NHRFinal.csv")
```

```
df_bronze_continent = spark.read.format("csv") \
    .option("header", "true") \
    .option("sep", ";") \
    .option("inferSchema", "true") \
    .load("/FileStore/tables/dataset/NHRContinent.csv")
```

```
▶ df_bronze_continent: pyspark.sql.dataframe.DataFrame = [Region: string, Continent: string]
```

```
▶ df_bronze_happiness: pyspark.sql.dataframe.DataFrame = [Country: string, Year: integer, 9 more fields]
```

27

```
df_bronze_happiness.printSchema()
df_bronze_happiness.show(5)
```

```
-- Region: string (nullable = true)
-- Happiness Rank: integer (nullable = true)
-- Happiness Score: string (nullable = true)
-- GDP: string (nullable = true)
-- Social Support: string (nullable = true)
-- Life Expectancy: string (nullable = true)
-- Freedom: string (nullable = true)
-- Generosity: string (nullable = true)
-- Corruption: string (nullable = true)
```

Country	Year	Region	Happiness Rank	Happiness Score	GDP	Social Support	Life Expectancy	Freedom	Generosity	Corruption
Afghanistan	2023	Southern Asia	137	1,859	0,645	0,000	0,087	0,000	0,093	0,059
Afghanistan	2017	Southern Asia	141	3,794	0,401	0,582	0,181	0,106	0,312	0,061
Afghanistan	2018	Southern Asia	145	3,632	0,332	0,537	0,255	0,085	0,191	0,036
Afghanistan	2022	Southern Asia	146	2,404	0,758	0,000	0,289	0,000	0,089	0,005
Afghanistan	2021	Southern Asia	149	2,523	0,370	0,000	0,126	0,000	0,122	0,010

only showing top 5 rows

28

```
#sParonizar as columnas
df_bronze_happiness = sanitize_column_names(df_bronze_happiness)
df_bronze_continent = sanitize_column_names(df_bronze_continent)
```

```
df_bronze_continent: pyspark.sql.dataframe.DataFrame = [region: string, continent: string]
df_bronze_happiness: pyspark.sql.dataframe.DataFrame = [country: string, year: integer ... 9 more fields]
```

29: Tratar dados na Silver

```
#Trocar vírgula por ponto nos números
from pyspark.sql.functions import regexp_replace, col, trim

# Lista das colunas que estão com números no formato string (com vírgula)
cols_to_clean = [
    "happiness_score", "gdp", "social_support",
    "life_expectancy", "freedom", "generosity", "corruption"
]

# Substituir vírgula por ponto e converter para float
for c in cols_to_clean:
    df_bronze_happiness = df_bronze_happiness.withColumn(
        c, regexp_replace(col(c), ",", ".").cast("float")
    )

# Padronizar colunas de texto
df_bronze_happiness = df_bronze_happiness.withColumn("country", trim(col("country")))
df_bronze_happiness = df_bronze_happiness.withColumn("region", trim(col("region")))
```

► df_bronze_happiness: pyspark.sql.dataframe.DataFrame = [country: string, year: integer ... 9 more fields]

30: Join com continent

```
# Join com a tabela de continentes
df_silver = df_bronze_happiness.join(
    df_bronze_continent,
    on="region",
    how="left"
)
```

► df_silver: pyspark.sql.dataframe.DataFrame = [region: string, country: string ... 10 more fields]

31: Validação join

```
df_silver.select("country", "region", "continent", "year", "happiness_score").show(5)
```

```
+-----+-----+-----+-----+
| country|      region|continent|year|happiness_score|
+-----+-----+-----+-----+
|Afghanistan|Southern Asia|      Asia|2023|          1.859|
|Afghanistan|Southern Asia|      Asia|2017|          3.794|
|Afghanistan|Southern Asia|      Asia|2018|          3.632|
|Afghanistan|Southern Asia|      Asia|2022|          2.404|
|Afghanistan|Southern Asia|      Asia|2021|          2.523|
+-----+-----+-----+-----+
only showing top 5 rows
```


32: Salvar como tabela Silver

```
# Salvar como Delta
df_silver.write.format("delta").mode("overwrite").save("/mnt/silver/happiness")

# Registrar a tabela no catálogo
spark.sql("DROP TABLE IF EXISTS silver_happiness")
spark.sql("""
    CREATE TABLE silver_happiness
    USING DELTA
    LOCATION '/mnt/silver/happiness'
""")
```

Out[185]: DataFrame[]

33: Criar dim_país


```
# Dimensão País – dim_país
#Contém informações geográficas para análise por país, região e continente. Cada país recebe um ID único (id_país) para ser usado como chave na tabela fato.
# Colunas:id_país (PK), country, region, continent

from pyspark.sql.functions import monotonically_increasing_id

# Selecionar apenas colunas geográficas distintas
dim_país = df_silver.select("country", "region", "continent").dropDuplicates()

# Criar coluna de ID único para cada país
dim_país = dim_país.withColumn("id_país", monotonically_increasing_id())

# Visualizar
dim_país.orderBy("country").show(10, truncate=False)
```

▶  dim_país: pyspark.sql.dataframe.DataFrame = [country: string, region: string ... 2 more fields]

country	region	continent	id_país
Afghanistan	Southern Asia	Asia	30
Albania	Central and Eastern Europe	Europe	25
Algeria	Middle East and Northern Africa	Middle East and Northern Africa	109
Angola	Sub-Saharan Africa	Africa	98
Argentina	Latin America and Caribbean	Latin America and Caribbean	106
Armenia	Central and Eastern Europe	Europe	138
Australia	Australia and New Zealand	Oceania	57
Austria	Western Europe	Europe	68
Azerbaijan	Central and Eastern Europe	Europe	66
Bahrain	Middle East and Northern Africa	Middle East and Northern Africa	142

34: Salvar dim_país como tabela Delta

```
dim_pais.write.format("delta").mode("overwrite").save("/mnt/gold/dim_pais")

spark.sql("DROP TABLE IF EXISTS dim_pais")
spark.sql("""
    CREATE TABLE dim_pais
    USING DELTA
    LOCATION '/mnt/gold/dim_pais'
""")
```

Out[187]: DataFrame[]

35: Criar dim_tempo

```
# Dimensão Tempo – dim_tempo
#Contém os anos em que foram registrados os dados, com um identificador único (id_tempo).
# Colunas: id_tempo(PK), year

# Selecionar anos únicos
dim_tempo = df_silver.select("year").dropDuplicates().orderBy("year")

# Adicionar id_tempo
dim_tempo = dim_tempo.withColumn("id_tempo", monotonically_increasing_id())

# Visualizar
dim_tempo.show()
```

▸  dim_tempo: pyspark.sql.dataframe.DataFrame = [year: integer, id_tempo: long]

```
+---+-----+
|year|id_tempo|
+---+-----+
|2015|      0|
|2016|      1|
|2017|      2|
|2018|      3|
|2019|      4|
|2020|      5|
|2021|      6|
|2022|      7|
|2023|      8|
+---+-----+
```

36: Salvar dim_tempo como tabela Delta

```
dim_tempo.write.format("delta").mode("overwrite").save("/mnt/gold/dim_tempo")

spark.sql("DROP TABLE IF EXISTS dim_tempo")
spark.sql("""
    CREATE TABLE dim_tempo
    USING DELTA
    LOCATION '/mnt/gold/dim_tempo'
""")
```

Out[189]: DataFrame[]

37: Criar fato_felicidade

```
# Join com dim_pais
df_fato = df_silver.join(
    dim_pais,
    on=["country", "region", "continent"],
    how="left"
)

# Join com dim_tempo
df_fato = df_fato.join(
    dim_tempo,
    on="year",
    how="left"
)

# Selecionar colunas da fato
fato_felicidade = df_fato.select(
    "id_pais", "id_tempo",
    "happiness_score",
    "happiness_rank",
    "gdp",
    "social_support",
    "life_expectancy",
    "freedom",
    "generosity",
    "corruption"
)

# Visualizar
fato_felicidade.show(5)
```

- df_fato: pyspark.sql.dataframe.DataFrame = [year: integer, country: string ... 12 more fields]
- fato_felicidade: pyspark.sql.dataframe.DataFrame = [id_pais: long, id_tempo: long ... 8 more fields]

id_pais	id_tempo	happiness_score	happiness_rank	gdp	social_support	life_expectancy	freedom	generosity	corruption
30	8	1.859	137	0.645	0.0	0.087	0.0	0.093	0.059
30	2	3.794	141	0.401	0.582	0.181	0.106	0.312	0.061
30	3	3.632	145	0.332	0.537	0.255	0.085	0.191	0.036
30	7	2.404	146	0.758	0.0	0.289	0.0	0.089	0.005
30	6	2.523	149	0.37	0.0	0.126	0.0	0.122	0.01

only showing top 5 rows

Tabela Fato — fato_felicidade

Tabela central do modelo estrela. Contém os indicadores de felicidade associados a um país e um ano, usando chaves para as dimensões.

Colunas:

- id_pais (FK)
- id_tempo (FK)
- happiness_score
- happiness_rank
- gdp
- social_support
- life_expectancy
- freedom
- generosity
- corruption

39: Salvar fato_felicidade como tabela Delta

```
fato_felicidade.write.format("delta").mode("overwrite").save("/mnt/gold/fato_felicidade")

spark.sql("DROP TABLE IF EXISTS fato_felicidade")
spark.sql("""
CREATE TABLE fato_felicidade
USING DELTA
LOCATION '/mnt/gold/fato_felicidade'
""")
```

Out[191]: DataFrame[]

40: Views temporárias

```
# Criar views temporárias
spark.table("fato_felicidade").createOrReplaceTempView("vw_fato_felicidade")
spark.table("dim_pais").createOrReplaceTempView("vw_dim_pais")
spark.table("dim_tempo").createOrReplaceTempView("vw_dim_tempo")
```

Finalização da Camada Silver

Dados limpos e enriquecidos com a informação de continente, realizado:

- join entre bronze_happiness e bronze_continent usando a coluna `region`
- verificação de consistência dos dados após o join
- salvamento da tabela `silver_happiness` no formato Delta
- Modelo dimensional (estrela)
- Criado o modelo dimensional estrela:

Fato: fato_felicidade — Contém os indicadores por país e ano

Dimensões:

dim_pais — Informações geográficas: país, região, continente

dim_tempo — Informações temporais: ano, possíveis agregações futuras

🌟 Camada Gold

Dados estão prontos para análise, última etapa do pipeline de dados tem como objetivo as análises exploratórias sobre o índice de felicidade mundial. Realiza-se:

- Consultas SQL
- Criação de visualizações e gráficos
- Interpretação de padrões globais
- Geração de insights para apoiar decisões baseadas em dados

As análises realizadas na Gold buscam responder 14 perguntas de negócio, abordando:

- Comparações entre países e continentes
- Evolução temporal da felicidade
- Fatores que mais influenciam a felicidade (PIB, suporte social, expectativa de vida, etc.)
- Destaques específicos como o Brasil e a América Latina



Perguntas

São 14 perguntas exploradas com base no dataset World Happiness Report (2015–2023), utilizando SQL e modelagem dimensional.

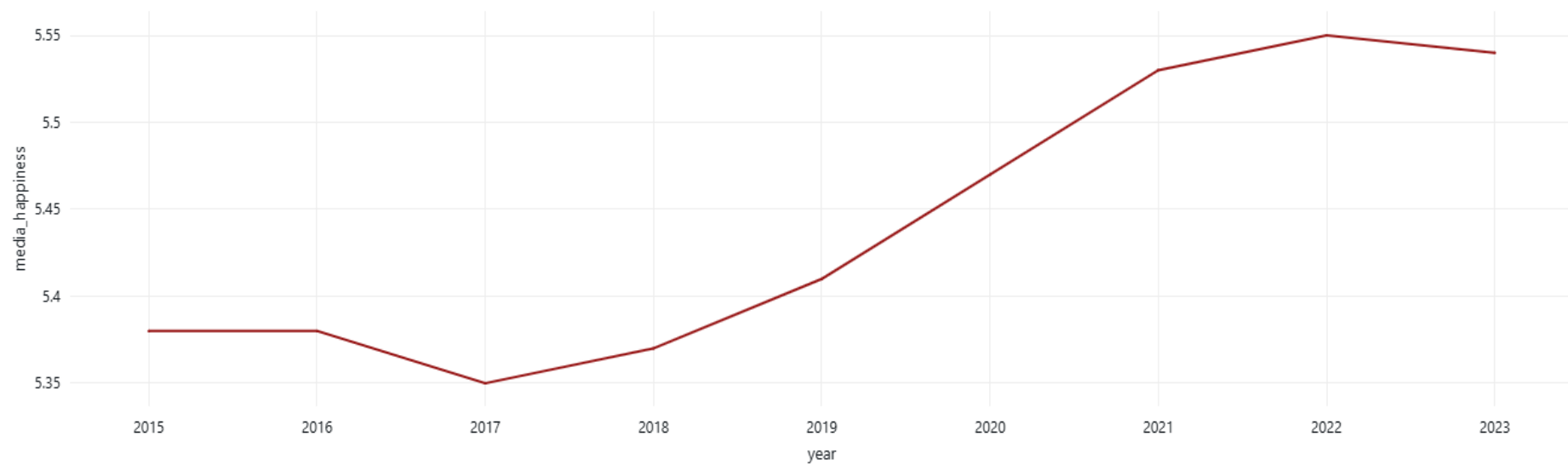
Nº	Pergunta	Métrica Principal
1	Qual a evolução da felicidade média no mundo por ano?	AVG(<code>happiness_score</code>) por ano
2	Comparativo de felicidade média por continente em 2023?	AVG(<code>happiness_score</code>) por continente
3	Top 10 países mais felizes em 2023?	TOP 10 <code>happiness_score</code>
4	Fatores mais presentes entre os países mais felizes?	Médias dos indicadores (<code>happiness_score</code> \geq 7)
5	Top 10 países menos felizes em 2023	Menores <code>happiness_score</code>
6	Quais fatores estão mais presentes entre os países menos felizes?	Médias dos indicadores (<code>happiness_score</code> \leq 4)
7	Quais são os 5 países menos felizes em cada ano?	<code>ROW_NUMBER()</code> por ano (<code>happiness_score</code> ASC)
8	Existe relação entre PIB per capita e felicidade?	Dispersão entre <code>gdp</code> e <code>happiness_score</code>
9	Como a expectativa de vida afeta a felicidade?	Dispersão entre <code>life_expectancy</code> e <code>happiness_score</code>
10	Quais países se destacam por generosidade ou suporte social?	AVG(<code>generosity</code>), AVG(<code>social_support</code>) por país
11	Correlação entre felicidade e GDP em 2023	<code>happiness_score</code> vs <code>gdp</code> (somente 2023)
12	Variação da felicidade no Brasil por ano?	Histórico de <code>happiness_score</code> do Brasil
13	Quais fatores estão mais presentes na felicidade do Brasil?	Médias dos indicadores do Brasil
14	Comparativo: Brasil vs América Latina e Caribe	Médias comparativas Brasil x América Latina

44: 1. Qual a evolução da felicidade média no mundo por ano?

```
%sql
SELECT t.year, ROUND(AVG(f.happiness_score), 2) AS media_happiness
FROM vw_fato_felicidade f
JOIN vw_dim_tempo t ON f.id_tempo = t.id_tempo
GROUP BY t.year
ORDER BY t.year;
```

► _sqldf: pyspark.sql.dataframe.DataFrame = [year: integer, media_happiness: double]

Table Visualização 1



9 rows

45: 2. Comparativo de felicidade média por continente em 2023?

```
%sql
SELECT p.continent, ROUND(AVG(f.happiness_score), 2) AS media_happiness
FROM vw_fato_felicidade f
JOIN vw_dim_pais p ON f.id_pais = p.id_pais
JOIN vw_dim_tempo t ON f.id_tempo = t.id_tempo
WHERE t.year = 2023
GROUP BY p.continent
ORDER BY media_happiness DESC;
```


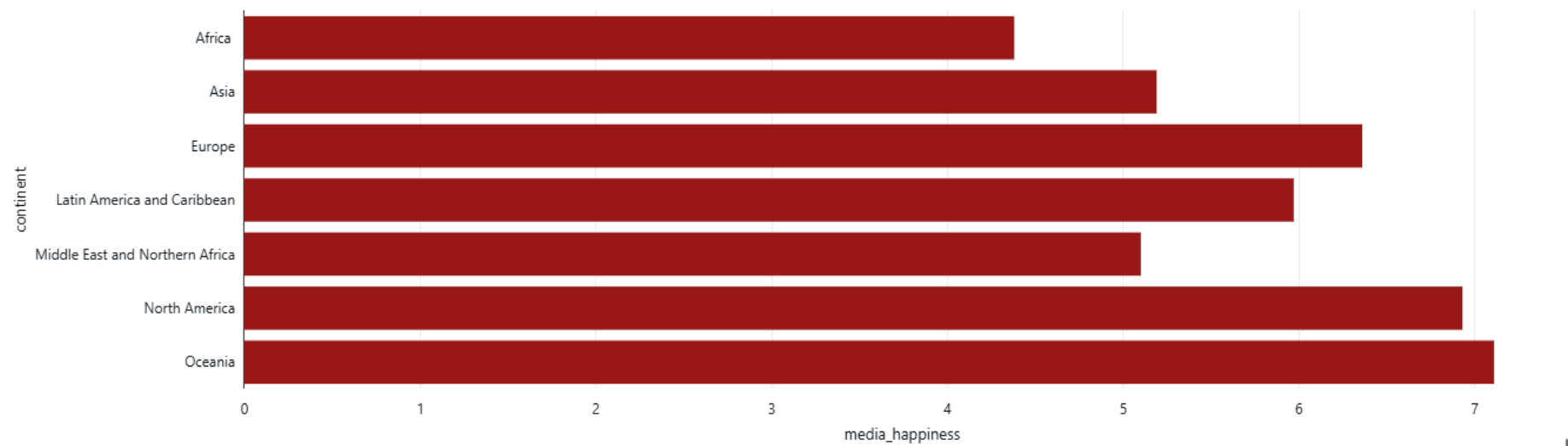
►  _sqldf: pyspark.sql.dataframe.DataFrame = [continent: string, media_happiness: double]

Table Visualização 1



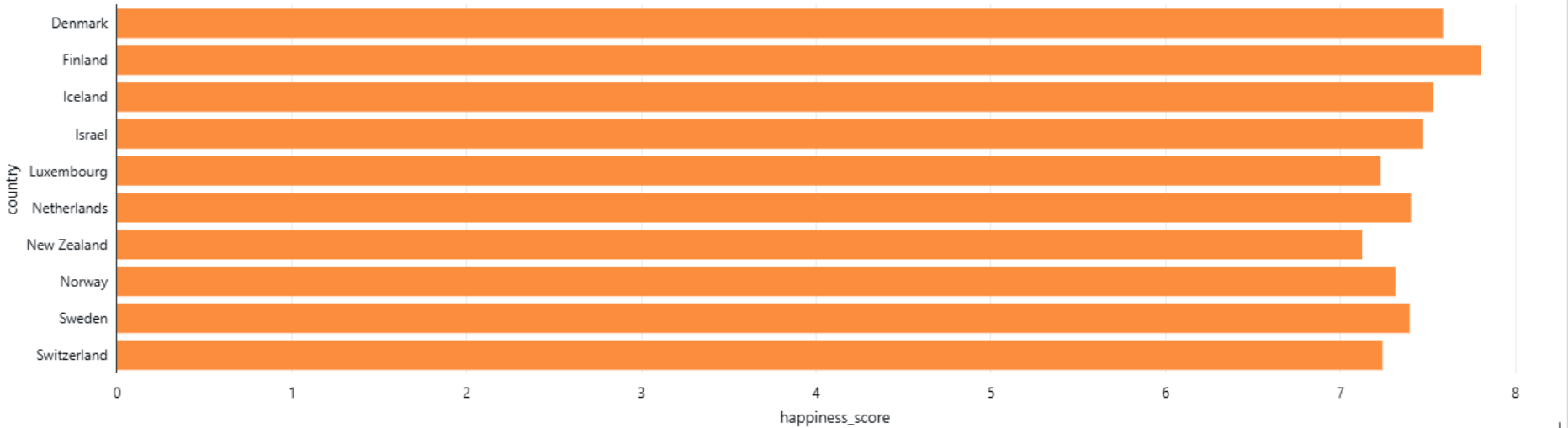
7 rows

46: 3. Top 10 países mais felizes em 2023?

```
%sql
SELECT p.country, f.happiness_score
FROM vw_fato_felicidade f
JOIN vw_dim_pais p ON f.id_pais = p.id_pais
JOIN vw_dim_tempo t ON f.id_tempo = t.id_tempo
WHERE t.year = 2023
ORDER BY f.happiness_score DESC
LIMIT 10;
```

_sqldf: pyspark.sql.dataframe.DataFrame = [country: string, happiness_score: float]

Table **Visualização 1**



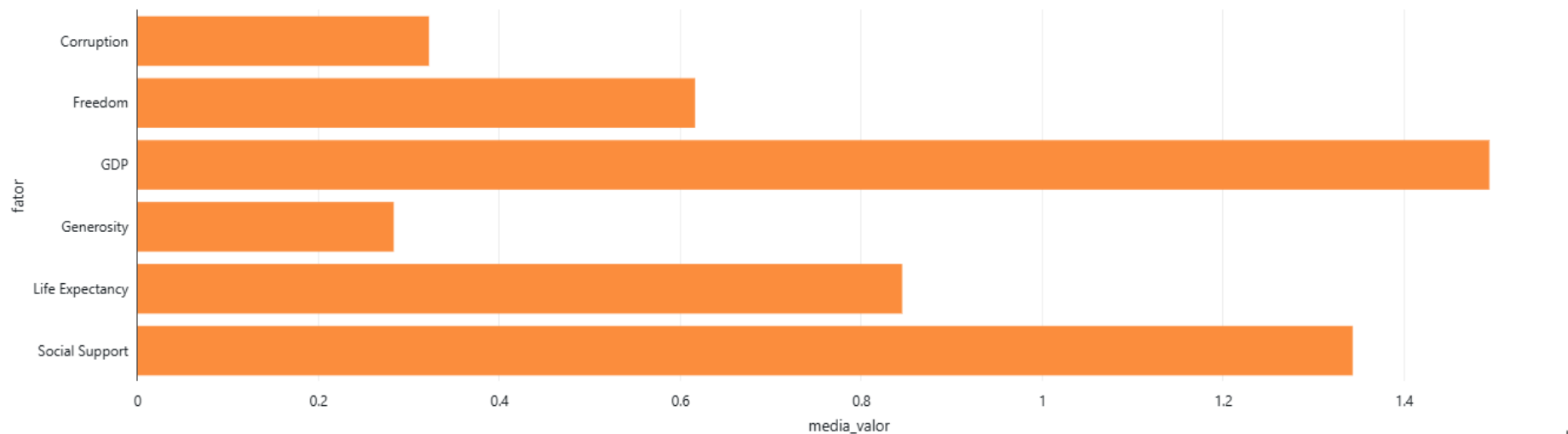
10 rows

47: 4. Fatores mais presentes entre os países mais felizes?

```
%sql
SELECT 'GDP' AS fator, ROUND(AVG(gdp), 3) AS media_valor FROM vw_fato_felicidade WHERE happiness_score >= 7
UNION ALL
SELECT 'Social Support', ROUND(AVG(social_support), 3) FROM vw_fato_felicidade WHERE happiness_score >= 7
UNION ALL
SELECT 'Life Expectancy', ROUND(AVG(life_expectancy), 3) FROM vw_fato_felicidade WHERE happiness_score >= 7
UNION ALL
SELECT 'Freedom', ROUND(AVG(freedom), 3) FROM vw_fato_felicidade WHERE happiness_score >= 7
UNION ALL
SELECT 'Generosity', ROUND(AVG(generosity), 3) FROM vw_fato_felicidade WHERE happiness_score >= 7
UNION ALL
SELECT 'Corruption', ROUND(AVG(corruption), 3) FROM vw_fato_felicidade WHERE happiness_score >= 7;
```

► _sqldf: pyspark.sql.dataframe.DataFrame = [fator: string, media_valor: double]

Table Visualização 1



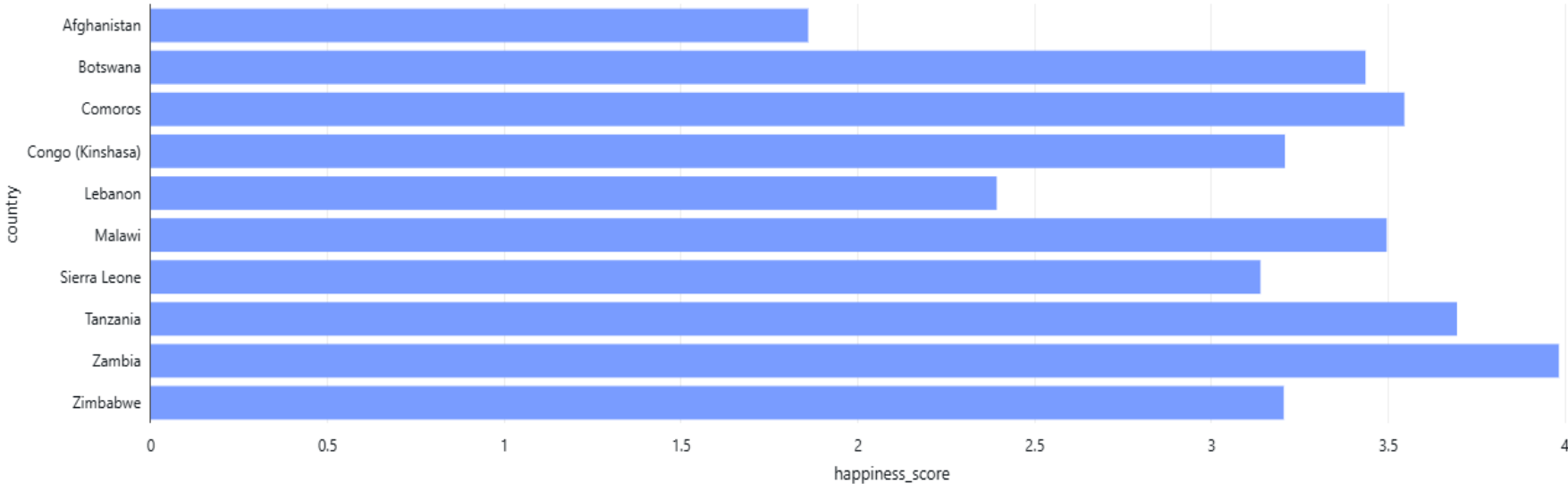
6 rows

48: 5. Top 10 países menos felizes em 2023?

```
%sql
SELECT p.country, f.happiness_score
FROM vw_fato_felicidade f
JOIN vw_dim_pais p ON f.id_pais = p.id_pais
JOIN vw_dim_tempo t ON f.id_tempo = t.id_tempo
WHERE t.year = 2023
ORDER BY f.happiness_score ASC
LIMIT 10;
```

_sqlidf: pyspark.sql.dataframe.DataFrame = [country: string, happiness_score: float]

Table Visualização 1



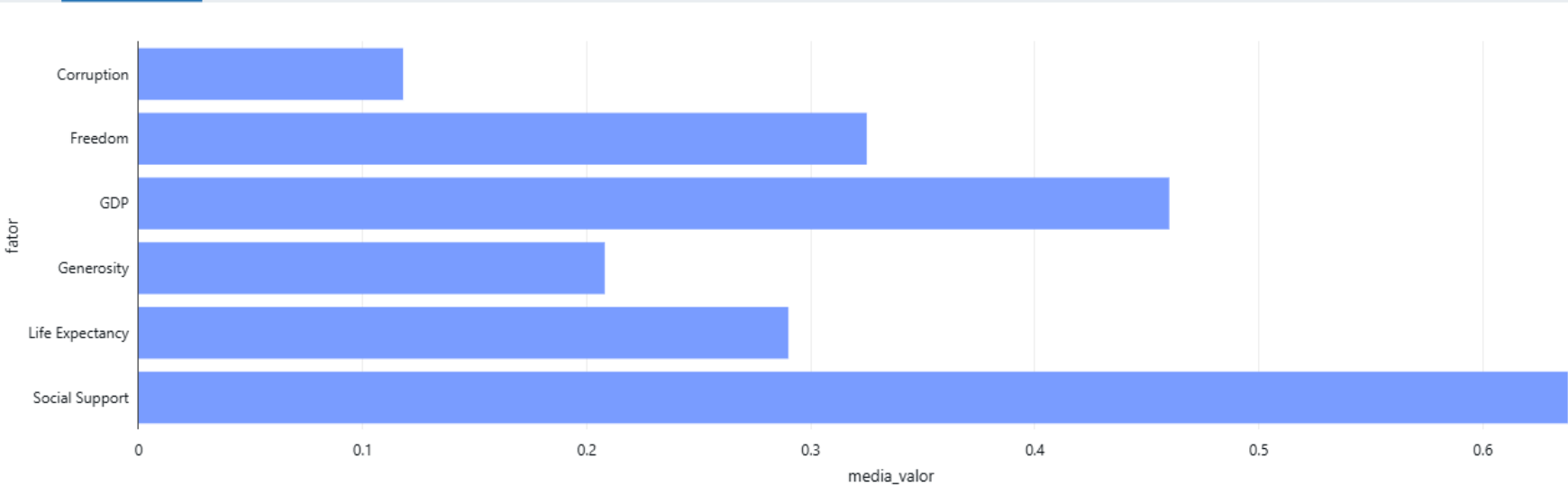
10 rows

49: 6. Quais fatores estão mais presentes entre os países menos felizes?

```
%sql
SELECT 'GDP' AS fator, ROUND(AVG(gdp), 3) AS media_valor FROM vw_fato_felicidade WHERE happiness_score <= 4
UNION ALL
SELECT 'Social Support', ROUND(AVG(social_support), 3) FROM vw_fato_felicidade WHERE happiness_score <= 4
UNION ALL
SELECT 'Life Expectancy', ROUND(AVG(life_expectancy), 3) FROM vw_fato_felicidade WHERE happiness_score <= 4
UNION ALL
SELECT 'Freedom', ROUND(AVG(freedom), 3) FROM vw_fato_felicidade WHERE happiness_score <= 4
UNION ALL
SELECT 'Generosity', ROUND(AVG(generosity), 3) FROM vw_fato_felicidade WHERE happiness_score <= 4
UNION ALL
SELECT 'Corruption', ROUND(AVG(corruption), 3) FROM vw_fato_felicidade WHERE happiness_score <= 4;
```

_sqldf: pyspark.sql.dataframe.DataFrame = [fator: string, media_valor: double]

Table Visualização 1



6 rows

50: 7. Quais são os 5 países menos felizes em cada ano?

```
%sql
WITH países_menos_felizes AS (
  SELECT
    t.year,
    p.country,
    f.happiness_score,
    ROW_NUMBER() OVER (PARTITION BY t.year ORDER BY f.happiness_score ASC) AS posicao
  FROM vw_fato_felicidade f
  JOIN vw_dim_pais p ON f.id_pais = p.id_pais
  JOIN vw_dim_tempo t ON f.id_tempo = t.id_tempo
  WHERE f.happiness_score IS NOT NULL
)

SELECT year, country, happiness_score
FROM países_menos_felizes
WHERE posicao <= 5
ORDER BY year, happiness_score;
```


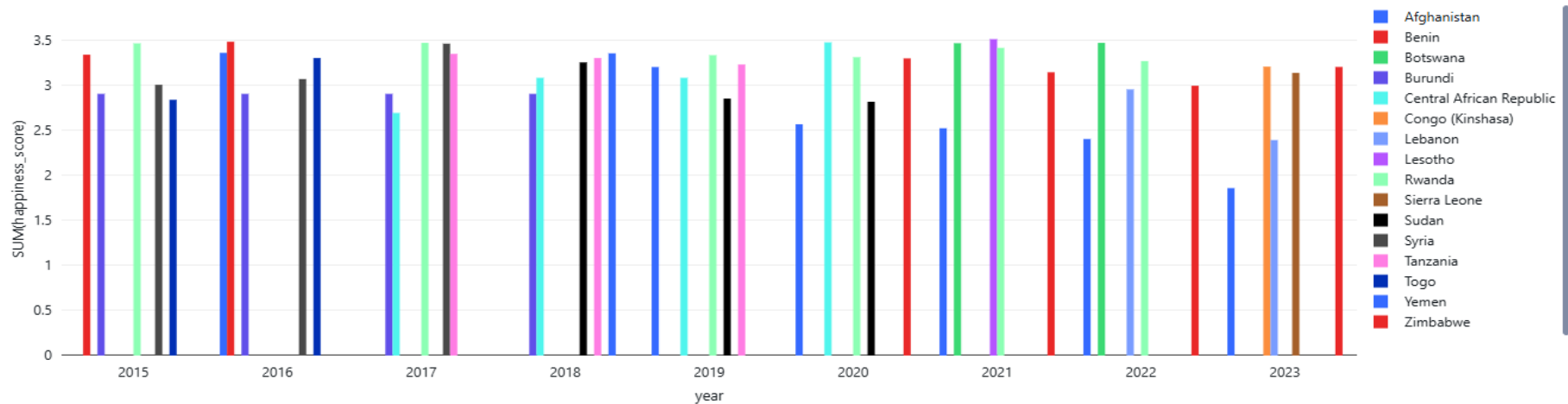
▸  _sqlidf: pyspark.sql.dataframe.DataFrame = [year: integer, country: string ... 1 more field]

Table Visualização 1



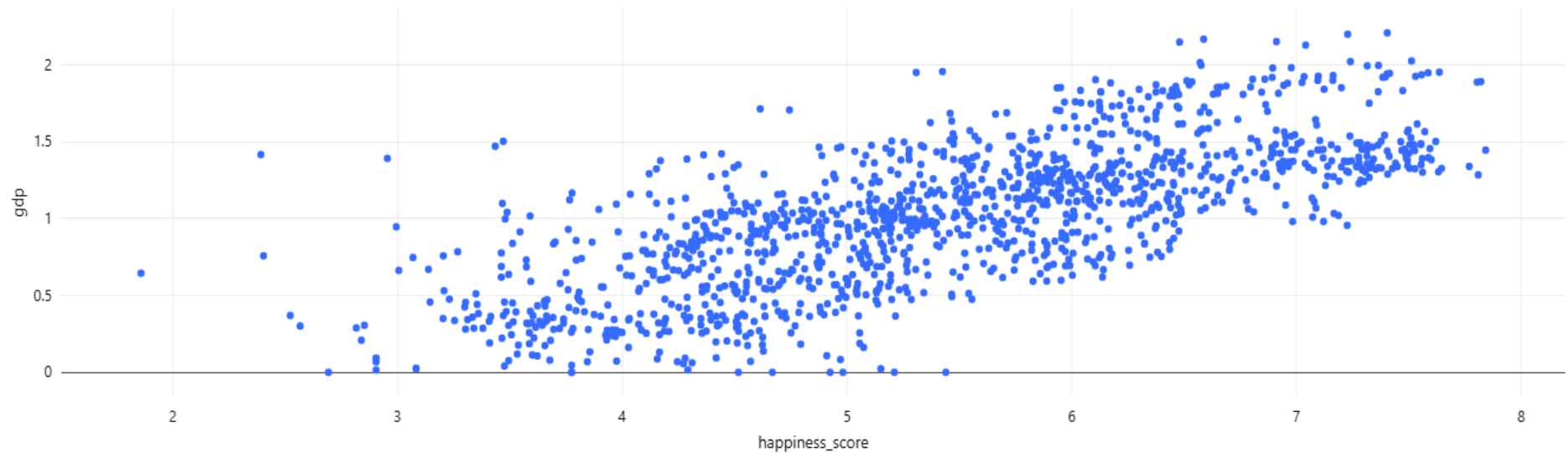
45 rows

51: 8. Existe relação entre PIB per capita e felicidade?

```
%sql
--Quanto maior o PIB per capita, maior tende a ser a felicidade?
SELECT f.happiness_score, f.gdp
FROM vw_fato_felicidade f
JOIN vw_dim_tempo t ON f.id_tempo = t.id_tempo
WHERE f.gdp IS NOT NULL AND f.happiness_score IS NOT NULL
```

▸  _sqldf: pyspark.sql.dataframe.DataFrame = [happiness_score: float, gdp: float]

Table Visualização 1



1,362 rows

52: 9. Como a expectativa de vida afeta a felicidade?

```
%sql
SELECT f.happiness_score, f.life_expectancy
FROM vw_fato_felicidade f
JOIN vw_dim_tempo t ON f.id_tempo = t.id_tempo
WHERE f.life_expectancy IS NOT NULL AND f.happiness_score IS NOT NULL
```

▸ `_sqldf: pyspark.sql.dataframe.DataFrame = [happiness_score: float, life_expectancy: float]`

Table **Visualização 1**



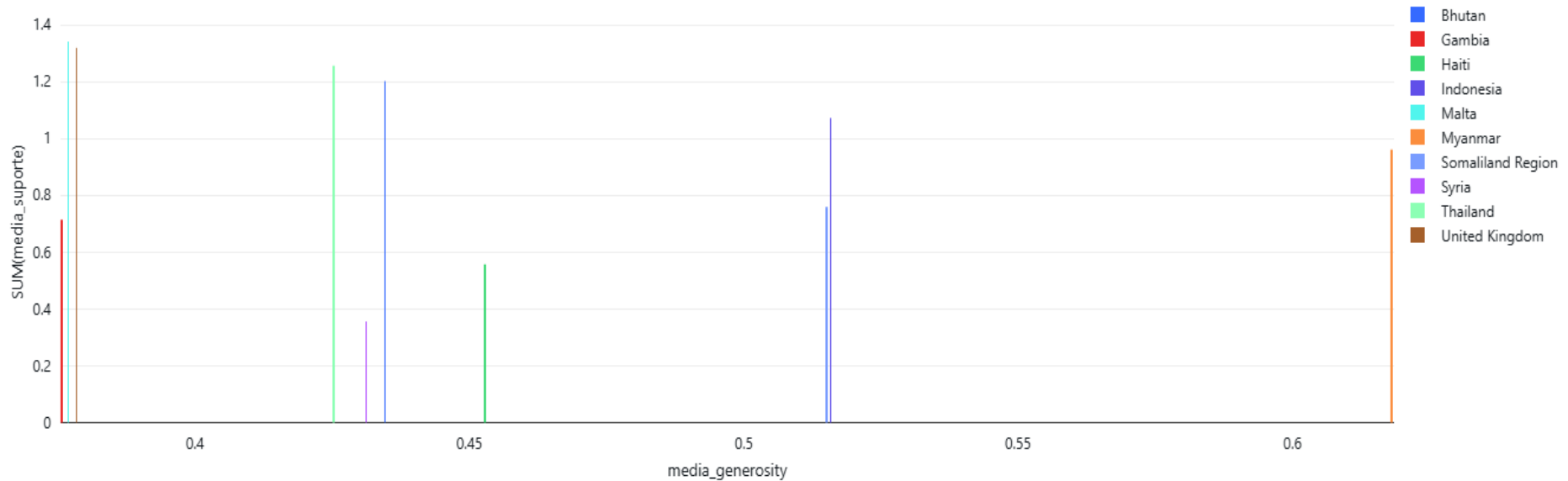
1,361 rows

53: 10. Quais países se destacam por generosidade ou suporte social?

```
%sql
SELECT p.country, ROUND(AVG(f.generosity), 3) AS media_generosity,
       ROUND(AVG(f.social_support), 3) AS media_suporte
FROM vw_fato_felicidade f
JOIN vw_dim_pais p ON f.id_pais = p.id_pais
GROUP BY p.country
ORDER BY media_generosity DESC
LIMIT 10;
```

▸ `_sqldf`: pyspark.sql.dataframe.DataFrame = [country: string, media_generosity: double ... 1 more field]

Table Visualização 1



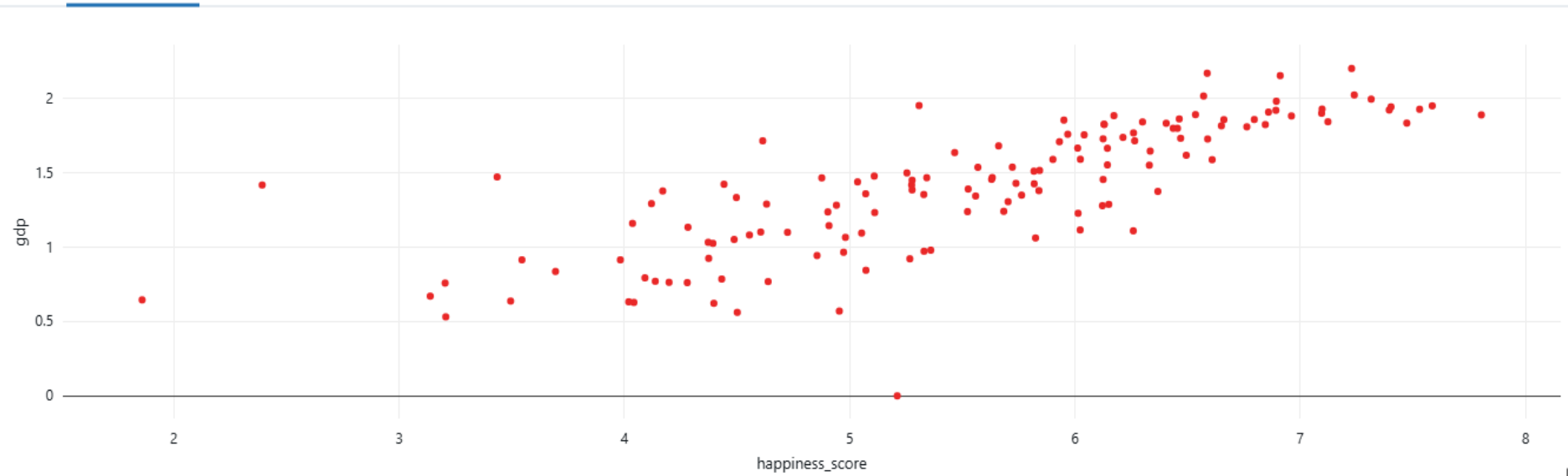
10 rows

54: 11. Correlação entre felicidade e GDP em 2023

```
%sql
SELECT f.happiness_score, f.gdp
FROM vw_fato_felicidade f
JOIN vw_dim_tempo t ON f.id_tempo = t.id_tempo
WHERE t.year = 2023
```

▸ _sqldf: pyspark.sql.dataframe.DataFrame = [happiness_score: float, gdp: float]

Table **Visualização 1**



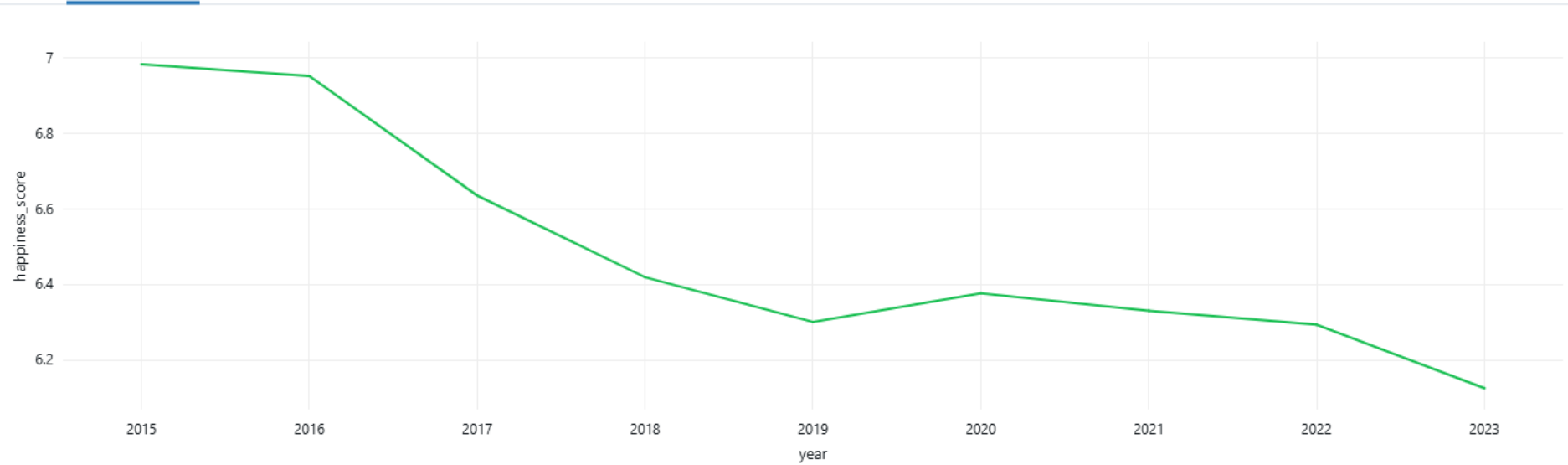
137 rows

55: 12. Variação da felicidade no Brasil por ano?

```
%sql
SELECT t.year, f.happiness_score
FROM vw_fato_felicidade f
JOIN vw_dim_pais p ON f.id_pais = p.id_pais
JOIN vw_dim_tempo t ON f.id_tempo = t.id_tempo
WHERE p.country = 'Brazil'
ORDER BY t.year;
```

►  _sqldf: pyspark.sql.dataframe.DataFrame = [year: integer, happiness_score: float]

Table **Visualização 1**



9 rows

56: 13. Quais fatores estão mais presentes na felicidade do Brasil?

```
%sql
SELECT
  'GDP' AS fator, ROUND(AVG(gdp), 3) AS media_valor
FROM vw_fato_felicidade f
JOIN vw_dim_pais p ON f.id_pais = p.id_pais
WHERE p.country = 'Brazil'

UNION ALL
SELECT 'Social Support', ROUND(AVG(social_support), 3)
FROM vw_fato_felicidade f JOIN vw_dim_pais p ON f.id_pais = p.id_pais
WHERE p.country = 'Brazil'

UNION ALL
SELECT 'Life Expectancy', ROUND(AVG(life_expectancy), 3)
FROM vw_fato_felicidade f JOIN vw_dim_pais p ON f.id_pais = p.id_pais
WHERE p.country = 'Brazil'

UNION ALL
SELECT 'Freedom', ROUND(AVG(freedom), 3)
FROM vw_fato_felicidade f JOIN vw_dim_pais p ON f.id_pais = p.id_pais
WHERE p.country = 'Brazil'

UNION ALL
SELECT 'Generosity', ROUND(AVG(generosity), 3)
FROM vw_fato_felicidade f JOIN vw_dim_pais p ON f.id_pais = p.id_pais
WHERE p.country = 'Brazil'

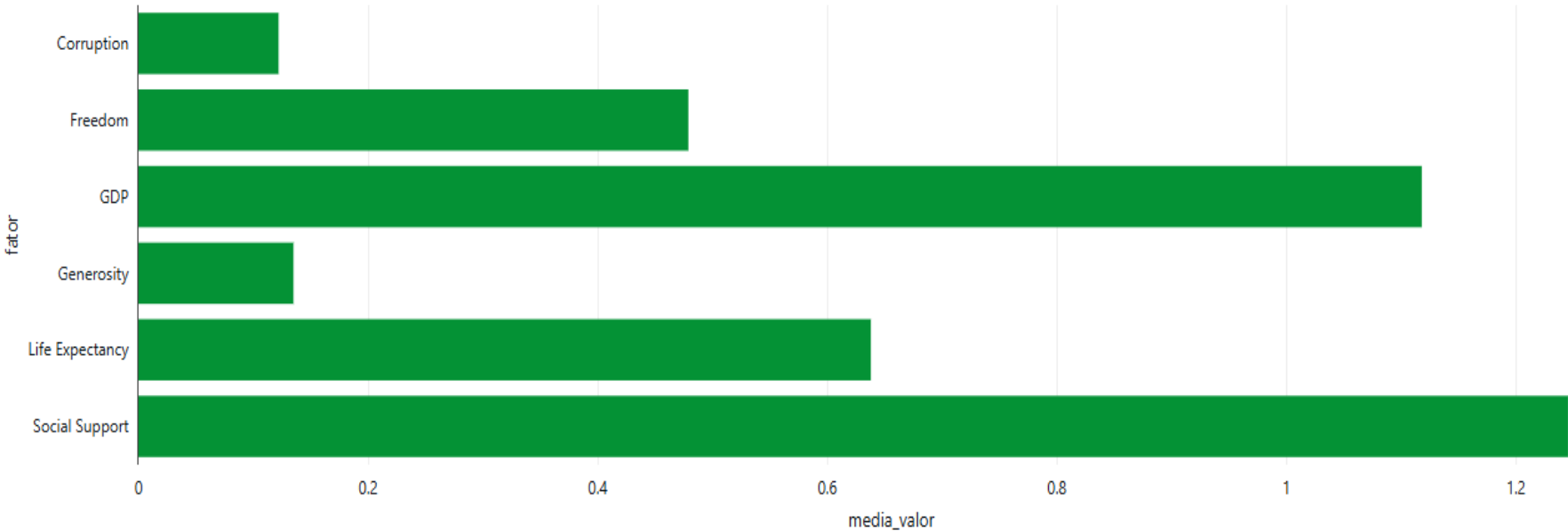
UNION ALL
SELECT 'Corruption', ROUND(AVG(corruption), 3)
FROM vw_fato_felicidade f JOIN vw_dim_pais p ON f.id_pais = p.id_pais
WHERE p.country = 'Brazil';
```

►  _sqldf: pyspark.sql.dataframe.DataFrame = [fator: string, media_valor: double]

Pipeline_World Happiness (Python)

_sqldf: pyspark.sql.dataframe.DataFrame = [fator: string, media_valor: double]

Table Visualização 1

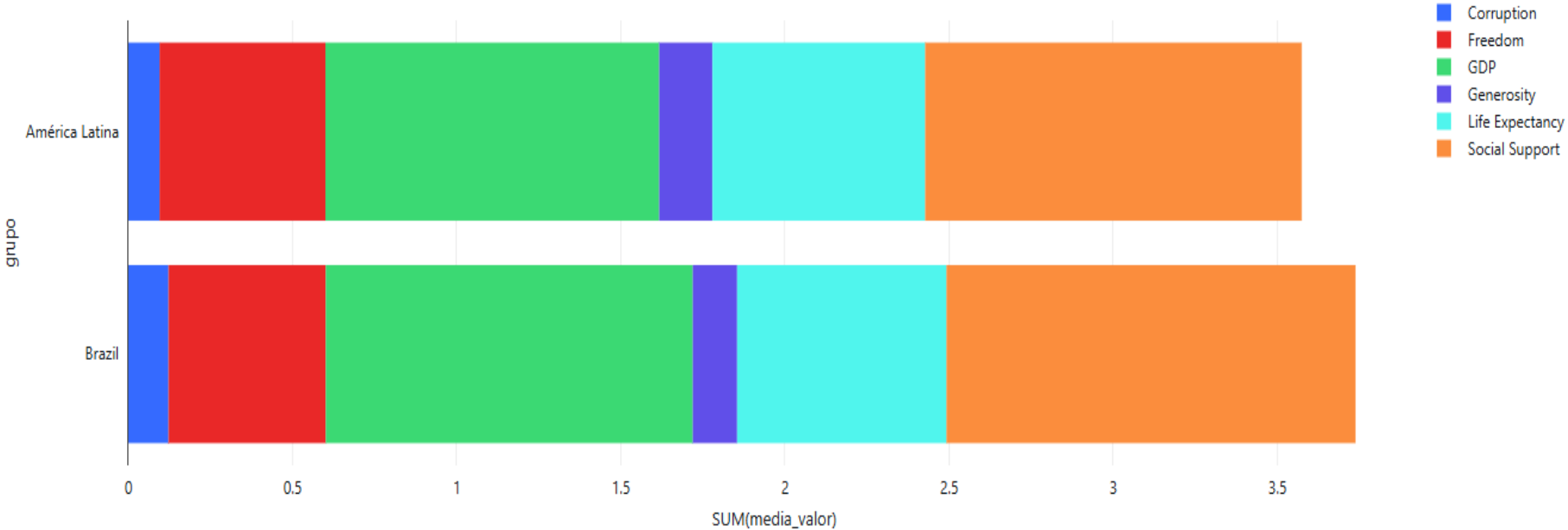


6 rows

57: 14. Comparativo Brasil vs América Latina e Caribe

```
%sql
-- MÉDIAS DO BRASIL
SELECT 'Brasil' AS grupo, 'GDP' AS fator, ROUND(AVG(gdp), 3) AS media_valor
FROM vw_fato_felicidade f
JOIN vw_dim_pais p ON f.id_pais = p.id_pais
WHERE p.country = 'Brazil'
UNION ALL
SELECT 'Brasil', 'Social Support', ROUND(AVG(social_support), 3)
FROM vw_fato_felicidade f JOIN vw_dim_pais p ON f.id_pais = p.id_pais WHERE p.country = 'Brazil'
UNION ALL
SELECT 'Brasil', 'Life Expectancy', ROUND(AVG(life_expectancy), 3)
FROM vw_fato_felicidade f JOIN vw_dim_pais p ON f.id_pais = p.id_pais WHERE p.country = 'Brazil'
UNION ALL
SELECT 'Brasil', 'Freedom', ROUND(AVG(freedom), 3)
FROM vw_fato_felicidade f JOIN vw_dim_pais p ON f.id_pais = p.id_pais WHERE p.country = 'Brazil'
UNION ALL
SELECT 'Brasil', 'Generosity', ROUND(AVG(generosity), 3)
FROM vw_fato_felicidade f JOIN vw_dim_pais p ON f.id_pais = p.id_pais WHERE p.country = 'Brazil'
UNION ALL
SELECT 'Brasil', 'Corruption', ROUND(AVG(corruption), 3)
FROM vw_fato_felicidade f JOIN vw_dim_pais p ON f.id_pais = p.id_pais WHERE p.country = 'Brazil'

-- MÉDIAS DA AMÉRICA LATINA
UNION ALL
SELECT 'América Latina', 'GDP', ROUND(AVG(gdp), 3)
FROM vw_fato_felicidade f JOIN vw_dim_pais p ON f.id_pais = p.id_pais
WHERE p.region = 'Latin America and Caribbean'
UNION ALL
SELECT 'América Latina', 'Social Support', ROUND(AVG(social_support), 3)
FROM vw_fato_felicidade f JOIN vw_dim_pais p ON f.id_pais = p.id_pais
WHERE p.region = 'Latin America and Caribbean'
UNION ALL
SELECT 'América Latina', 'Life Expectancy', ROUND(AVG(life_expectancy), 3)
FROM vw_fato_felicidade f JOIN vw_dim_pais p ON f.id_pais = p.id_pais
WHERE p.region = 'Latin America and Caribbean'
UNION ALL
SELECT 'América Latina', 'Freedom', ROUND(AVG(freedom), 3)
FROM vw_fato_felicidade f JOIN vw_dim_pais p ON f.id_pais = p.id_pais
WHERE p.region = 'Latin America and Caribbean'
UNION ALL
SELECT 'América Latina', 'Generosity', ROUND(AVG(generosity), 3)
FROM vw_fato_felicidade f JOIN vw_dim_pais p ON f.id_pais = p.id_pais
WHERE p.region = 'Latin America and Caribbean'
UNION ALL
SELECT 'América Latina', 'Corruption', ROUND(AVG(corruption), 3)
FROM vw_fato_felicidade f JOIN vw_dim_pais p ON f.id_pais = p.id_pais
WHERE p.region = 'Latin America and Caribbean';
```



Principais Insights do Pipeline de Dados da Felicidade Mundial

Com base nas análises desenvolvidas a partir do pipeline de dados sobre a felicidade mundial (World Happiness Report 2015–2023), foi possível chegar às seguintes conclusões:

- A média global de felicidade apresentou pequenas variações ao longo dos anos, com destaque para a leve queda em 2017.
- A Europa, North America, Oceania apresentam os maiores índices médios de felicidade em 2023, a África e a Ásia Meridional concentram os menores índices.
- Top 10 Países Mais Felizes (2023), Finlândia, Dinamarca e Islândia lideram consistentemente o ranking. Esses países apresentam forte suporte social, alta expectativa de vida e baixo índice de corrupção.
- Top 10 Países Menos Felizes (2023), Afeganistão, Líbano e Zimbábue estão entre os menos felizes. Os fatores mais baixos observados foram o PIB per capita, suporte social e liberdade.
- Fatores mais presentes entre os países mais felizes: Os principais são suporte social elevado, liberdade para decisões e expectativa de vida alta.
- Fatores entre os países menos felizes: apresentaram baixos níveis de PIB, expectativa de vida e suporte social.
- Países menos felizes por ano: ao longo dos anos, certos países apareceram repetidamente entre os menos felizes, revelando estabilidade negativa, destacam-se: Afghanistan, Benin, Botswana, Burundi, Lesotho, Syria.
- PIB vs Felicidade: identifica-se uma correlação positiva entre PIB per capita e felicidade, países mais ricos tendem a ser mais felizes.
- Expectativa de Vida vs Felicidade: a expectativa de vida tem relação direta com o índice de felicidade. Populações com maior longevidade relatam maior bem-estar.
- Generosidade e Suporte Social: países como Myanmar, Indonésia e Nova Zelândia apresentaram altos índices de generosidade, suporte social foi mais elevado em países da Europa.
- Correlação entre GDP e Felicidade (2023): observa-se um padrão claro de crescimento do índice de felicidade conforme o PIB per capita aumenta.
- Felicidade no Brasil ao longo do tempo: o Brasil apresenta queda ao longo dos anos de pandemia e pós pandemia de Covid, com a pior queda do período pós pandemia e 2023.
- Fatores presentes na felicidade do Brasil: os principais fatores no Brasil são o suporte social e PIB, enquanto liberdade e percepção de corrupção aparecem com menor pontuação.
- Comparativo: Brasil vs América Latina: o Brasil apresenta desempenho semelhante à média da América Latina em vários indicadores, destacando-se positivamente suporte social e negativamente em percepção de corrupção.

Autoavaliação do Projeto

Ao longo do desenvolvimento deste projeto, foi possível aplicar de forma prática os conceitos de pipeline de dados, organização por camadas (Bronze, Silver e Gold), modelagem dimensional e análises exploratórias com foco em resolução de problemas de negócio.

As análises demonstram que o pipeline foi eficaz na transformação dos dados em conhecimento, permitindo interpretações ricas sobre a felicidade e bem estar global e regional.



Objetivos Alcançados

- Construção do pipeline completo no ambiente Databricks Community Edition, com todas as camadas operacionais (Bronze → Silver → Gold).
- Leitura, limpeza e padronização de dados reais, provenientes de um dataset público da plataforma Kaggle.
- Modelagem estrela, com separação clara entre fatos e dimensões, permitindo consultas analíticas robustas.
- Criação de 14 perguntas de negócio relevantes que exploram aspectos sociais, econômicos e geográficos da felicidade mundial.
- Geração de visualizações no Databricks para facilitar a interpretação dos dados.



Desafios

- Dificuldades iniciais no uso do Databricks, especialmente para entender o funcionamento dos clusters, execução de notebooks e gestão dos arquivos no DBFS.
- Desconexão frequente do cluster por inatividade, exigindo reativação e carga dos blocos de código.
- Necessidade de lidar com problemas na leitura dos arquivos CSV, como separadores incorretos e conversão de tipos.
- Aprendizado gradual sobre como aplicar a modelagem dimensional na prática, incluindo criação de tabelas fato e dimensão com joins.
- A construção do processo de ETL exigiu buscar novos conhecimentos, especialmente sobre tratamento de dados com PySpark e a lógica por trás das camadas Bronze, Silver e Gold — o que trouxe muito aprendizado técnico e prático.
- No compartilhamento do notebook no Github, constatei que os gráficos não ficam visíveis e não consegui realizar a correção.
- Disponibilizado "pdf" do notebook para a visualização dos gráficos.
- É um aprendizado gradual

Atingimento dos Objetivos

Todos os objetivos traçados no início do trabalho foram atingidos.

- O pipeline foi construído com todas as suas camadas funcionais.
- Os dados foram tratados, enriquecidos e organizados em uma modelagem estrela.
- Todas as perguntas de negócio foram respondidas com base nos dados disponíveis.
- As visualizações permitiram compreender padrões e gerar insights relevantes sobre a felicidade mundial.

Reflexão Final

O projeto foi extremamente valioso e trouxe muito aprendizado — tanto no aspecto técnico quanto na organização de um raciocínio analítico orientado por dados.

Mesmo com os desafios, especialmente ligados ao uso da plataforma em nuvem, foi possível implementar todas as etapas do pipeline com sucesso. A construção do ETL, em especial, exigiu pesquisa e prática, resultando em aprendizado.

Através da modelagem e das análises, consegui obter respostas claras e fundamentadas para todas as perguntas de negócio propostas, extraindo insights significativos sobre os fatores que influenciam a felicidade ao redor do mundo.

Além disso, este projeto representou um reencontro com a programação, uma área pela qual tenho apreço e que me trouxe satisfação ao longo do desenvolvimento — algo que me motiva e inspira para os próximos passos profissionais. É o começo de um projeto de transição de carreira, com foco maior em dados, tecnologia e soluções analíticas