

# Engenharia de Dados



Rodrigo Batista de Souza  
1º Semestre/2025

# Sumário

<b>1 ) Introdução:</b> .....	2
Pipeline de Dados - Definição: .....	2
<b>2) Proposta:</b> .....	2
Plataforma para o projeto: .....	3
Premissas: .....	3
<b>3 ) Coleta:</b> .....	3
<b>4 ) Modelagem:</b> .....	5
<b>5 ) Processo ETL:</b> .....	6
Definição: .....	6
Camada Bronze: .....	6
Camada Silver: .....	8
Camada Gold: .....	11
<b>6 ) Métricas:</b> .....	15
Total de casos e mortes por continente: .....	15
Evolução de casos em um determinado país: .....	16
Ranking Top-10 de países com maior taxa de mortalidade: .....	17
<b>7 ) Dashboards:</b> .....	19
Relação de mortes e casos por continente: .....	20
Evolução de casos em um determinado país: .....	21
Ranking top-10 dos países com maior taxa de mortalidade: .....	23
<b>8 ) Autoavaliação:</b> .....	24
<b>9 ) Anexos:</b> .....	25
Catálogo de Dados: .....	25
Metadados das camadas: .....	25
Dicionário de Dados: .....	26
Fluxo de Dados: .....	26
Relacionamentos: .....	27
Criação dos clusters na Databricks: .....	28

# **MVP da sprint de Engenharia de Dados – 1º Semestre/2025**

## **1 ) Introdução:**

Este projeto tem por objetivo fazer a construção de um pipeline de dados utilizando tecnologias na nuvem e consolidar os conhecimentos adquiridos no estudo do módulo. O pipeline irá envolver as etapas de busca, coleta, modelagem, carga e análises dos dados imputados, de forma a tentar responder algumas perguntas previamente definidas, as quais serão úteis para a compreensão macro do projeto.

### **Pipeline de Dados - Definição:**

Um pipeline de dados consiste em um conjunto de processos onde são implementadas as etapas necessárias para mover os dados de sistemas de origem, transformar esses dados com base em requisitos predefinidos e armazená-los em um sistema de destino. Em outras palavras, inclui todas as etapas necessárias para que dados brutos (sem alterações ou com alterações mínimas na sua origem) sejam transformados em dados preparados para que os usuários interessados possam consumir as informações. E são estas premissas iniciais que vão delinear o desenvolvimento deste projeto.

## **2) Proposta:**

A proposta deste trabalho é elaborar toda a estrutura necessária para a compreensão e análise de dados utilizando o ambiente de computação em nuvem. Os dados utilizados como base são referentes às análises de dados de COVID extraídos do site *Our World in Data* (OWID) (<https://covid.ourworldindata.org/data/owid-covid-data.csv>), uma organização

não-governamental que ajuda no desenvolvimento de várias pesquisas ao redor do mundo.

### Plataforma para o projeto:

Para a construção do projeto, optou-se pelo uso da plataforma *Databricks Community Edition* (<https://community.cloud.databricks.com>). Trata-se de um ambiente totalmente gratuito, com algumas limitações, é verdade, em relação a outras plataformas já consolidadas e consagradas no mercado. No entanto, para este estudo, a plataforma mostrou-se ser bastante satisfatória para a execução de todos os passos necessários e, consequentemente, realizar o desenvolvimento do trabalho.

### Premissas:

A idealização deste projeto partiu de algumas perguntas que gostaria que fossem possíveis de serem respondidas tais como:

- Qual o número total de casos que aconteceram por continente?
- Como os casos evoluíram em um determinado país?
- Quais foram os 10 países que tiveram a maior taxa de mortalidade?

Tendo em mãos a resposta sobre estes questionamentos, objetivou-se, com isso, a abertura de opções para que se pudesse efetuar uma demonstração destes resultados, de forma a se ter uma melhor visualização do conteúdo e posterior compreensão da mensagem exposta. A forma escolhida foi a projeção de dashboards na forma de gráficos no próprio código, de uma maneira bem simples, direta e de tranquilo entendimento.

### **3 ) Coleta:**

O *dataset* de origem é proveniente do site *Our World in Data*, uma organização sem fins lucrativos que possui vasta base de dados sobre diversos temas sensíveis da sociedade tais como pobreza, doenças, mudanças

climáticas, guerras, entre outros temas do nosso cotidiano. Possui contribuições de várias pesquisas ao redor do mundo de pesquisadores atuantes em instituições renomadas como Oxford na Inglaterra, por exemplo. E a disseminação destes dados para conhecimento geral é a maior importância de sua existência. As informações contidas neste site são de boa procedência e, como possui uma grande quantidade de registros organizados, conseguimos demonstrar as informações com qualidade.

Os dados são disponibilizados em formato .csv e a ingestão destes para o trabalho foi feita através da própria plataforma Databricks Community Edition. A figura abaixo mostra como foi feito o processo da coleta dos arquivos para a plataforma Databricks.<sup>1</sup>

```
import pandas as pd
from io import StringIO
import requests

# URL dos dados (fonte original)
url = "https://covid.ourworldindata.org/data/owid-covid-data.csv"

# Baixar os dados diretamente para um DataFrame pandas
response = requests.get(url)
csv_data = StringIO(response.text)
pdf = pd.read_csv(csv_data)
```

Figura 1 – Coleta dos dados

Feita esta parte, manipularemos os dados usando uma metodologia conhecida como arquitetura em camadas (formato *Delta Lake*), detalhada mais à frente.

---

<sup>1</sup> O notebook completo com todos os passos do código está disponível no repositório do Github ([https://github.com/rodrigobsouza17/Engenharia\\_de\\_Dados/tree/main/MVP](https://github.com/rodrigobsouza17/Engenharia_de_Dados/tree/main/MVP)).

#### 4 ) Modelagem:

A modelagem escolhida para o projeto foi a chamada arquitetura em camadas, estrutura clássica muito utilizada na área de Engenharia de Dados.

As camadas são o resultado do fluxo de dados utilizado e servem, sobretudo, para tratarmos melhor os dados deste fluxo. Em cada camada, o dado é tratado de uma forma distinta, sendo trabalhado em níveis incrementais de qualidade. São divididas em:

- Bronze: Tabelas em formato bruto, com o máximo de fidelidade em relação ao dado original coletado
- Silver: Tabelas com estrutura e *schema* definidos, onde são realizadas tanto a limpeza quanto o enriquecimento dos dados
- Gold: Tabelas agregadas com métricas de interesse de acordo com a área de negócio e demais interessados.

A ilustração abaixo reflete a essência de como é o funcionamento das camadas.

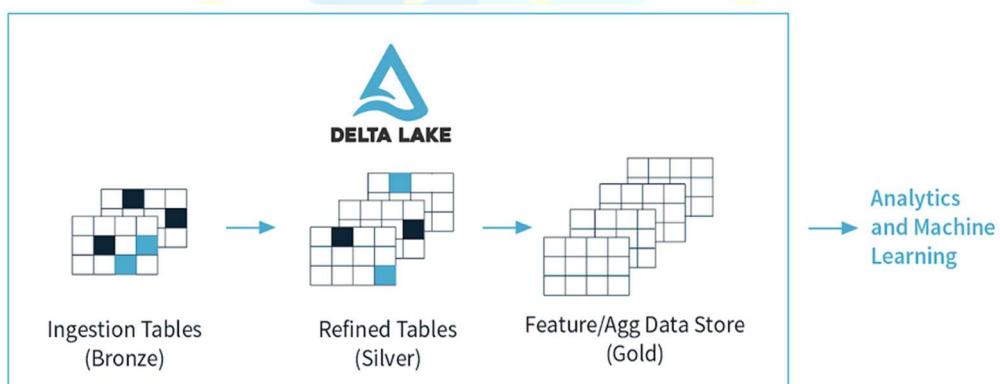


Figura 2 – Arquitetura em Camadas

O processo de criação destas camadas será detalhado logo mais adiante, na parte de processo ETL.

## 5 ) Processo ETL:

### Definição:

ETL é o acrônimo de três palavras em inglês que consistem nas etapas do processo. Elas são as seguintes:

- Extract (extração): É o passo em que extraímos as informações em seu formato bruto, com o máximo de fidelidade ao dado original coletado.
- Transformation (transformação): Passo onde realizamos a limpeza e transformação dos dados obtidos na extração.
- Load (carregamento): Passo onde organizamos os dados transformados de forma a estarem prontos para análises pelos interessados.

O processo de ETL é útil para, sobretudo, melhorar a governança, performance e confiabilidade dos dados. Ou seja, ele torna-se essencial para garantir que os dados fiquem organizados, seguros, eficientes e acessíveis para futuras tomadas de decisão.

A seguir, detalharemos os impactos da segregação em camadas no processo de ETL.

### Camada Bronze:

Também chamada de camada *Raw*. Conforme dito anteriormente, a camada bronze é o local onde armazenamos as tabelas em seu estado puro, ou seja, com o máximo de fidelidade possível dos dados originais. No processo de ETL, esta camada fará parte do passo de extração.

Logo abaixo, evidenciamos como foi feita a criação da camada bronze no código.

```
dbutils.fs.mkdirs("dbfs:/mnt/covid_data/bronze")
```

Figura 3 – Criação do diretório para a gravação da camada bronze na Databricks

```
# Converter para Spark DataFrame e salvar como formato Delta
bronze_path = "dbfs:/mnt/covid_data/bronze/owid-covid-data.delta"
spark_df = spark.createDataFrame(pdf)
spark_df.write.format("delta").mode("overwrite").save(bronze_path)

# Registrar tabela
spark.sql("""
CREATE TABLE IF NOT EXISTS bronze_covid_data
USING DELTA
LOCATION 'dbfs:/mnt/covid_data/bronze/owid-covid-data.delta'
""")
```

Figura 4 – Criação da camada bronze

Após a coleta dos dados, convertemos os dados para serem lidos pela Databricks e criamos a tabela usando comandos SQL.

Podemos confirmar que a tabela foi criada observando o Catálogo da Databricks, conforme abaixo.

The screenshot shows the Databricks interface with the sidebar menu open. The 'Catalog' option is selected. In the main area, under the 'Data' tab, there is a 'Databases' section with a dropdown set to 'default'. Below it is a 'Tables' section. A table named 'bronze\_covid\_data' is listed in the tables list. The interface has a dark theme with orange and grey accents.

Figura 5 – Tabela da Camada Bronze

Para saber se a tabela foi criada corretamente, optou-se por uma verificação adicional, por garantia.

# Verificar se os dados foram carregados corretamente ("Check")  
display(spark.sql("SELECT COUNT(\*) FROM bronze\_covid\_data"))  
display(spark.sql("SELECT \* FROM bronze\_covid\_data LIMIT 5"))

(6) Spark Jobs

Table +

	count(1)
1	429435

Table +

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed
1	AFG	Asia	Afghanistan	2020-01-05	0	0	
2	AFG	Asia	Afghanistan	2020-01-06	0	0	
3	AFG	Asia	Afghanistan	2020-01-07	0	0	
4	AFG	Asia	Afghanistan	2020-01-08	0	0	
5	AFG	Asia	Afghanistan	2020-01-09	0	0	

5 rows | 11.24s runtime Refreshed 2 minutes ago

Figuras 6 e 7 – Tabela da camada bronze

### Camada Silver:

A camada silver é o local onde realizamos a limpeza e transformação dos dados obtidos da camada bronze. Em outras palavras, podemos, por exemplo, tratar dados inconsistentes, filtrar nulos e padronizar formatos.

A seguir, evidenciamos a criação da camada silver via código.

```
dbutils.fs.mkdirs("dbfs:/mnt/covid_data/silver")
```

Figura 8 – Criação do diretório para a gravação

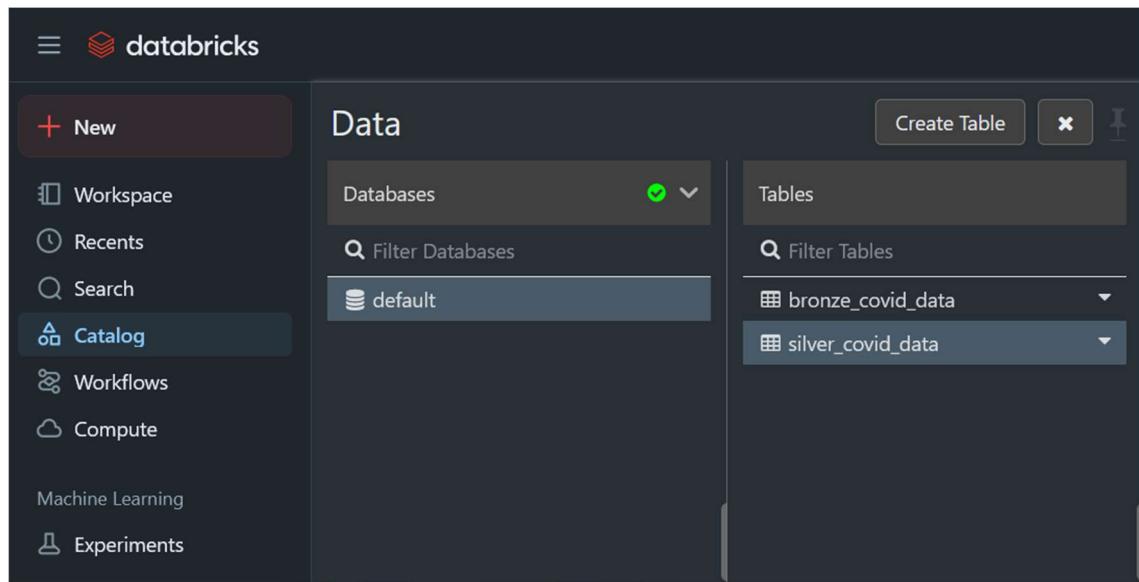
```
# Ler da tabela bronze
silver_df = spark.sql("""
SELECT
    iso_code AS country_code,
    location AS country_name,
    continent,
    to_date(date) AS date,
    total_cases,
    new_cases,
    total_deaths,
    new_deaths,
    population,
    gdp_per_capita,
    human_development_index
FROM bronze_covid_data
WHERE continent IS NOT NULL
"""\

# Escrever na camada silver
silver_path = "dbfs:/mnt/covid_data/silver/covid_data.delta"
silver_df.write.format("delta").mode("overwrite").save(silver_path)

# Registrar tabela
spark.sql("""
CREATE TABLE IF NOT EXISTS silver_covid_data
USING DELTA
LOCATION 'dbfs:/mnt/covid_data/silver/covid_data.delta'
""")
```

Figuras 9 e 10 – Criação da tabela na camada silver

E também observamos, com isso, a criação da tabela lá no catálogo.



The screenshot shows the Databricks interface with the sidebar open. The 'Catalog' option is selected. In the main area, under 'Databases', the 'default' database is selected. Under 'Tables', two tables are listed: 'bronze\_covid\_data' and 'silver\_covid\_data'. The 'silver\_covid\_data' table is highlighted with a blue selection bar.

Figura 11 – Tabela da camada silver

As verificações de qualidade são muito importantes nessa fase do processo, para garantir que os dados foram de fato transformados e estão prontos para *analytics*.

Abaixo evidenciamos como poderia ser o resultado, por exemplo, de uma verificação de qualidade executada nesta parte do processo.

```
==== Verificação de Qualidade para silver_covid_data ===
Total de registros: 402910
Valores nulos em country_code: 0 (0.00%)
Valores nulos em country_name: 0 (0.00%)
Valores nulos em date: 0 (0.00%)
Valores nulos em new_cases: 12839 (3.19%)
Registros antes de 2019-12-01: 0
Registros após 2023-12-31: 51055
Registros sem continente: 0 (deveria ser 0)
Possíveis duplicatas (mesmo país+data): 1408
```

Figura 12 – Exemplo de como poderia ser uma verificação de qualidade<sup>2</sup>

---

<sup>2</sup> O notebook completo com todos os passos do código está disponível no repositório do Github ([https://github.com/rodrigobsouza17/Engenharia\\_de\\_Dados/tree/main/MVP](https://github.com/rodrigobsouza17/Engenharia_de_Dados/tree/main/MVP)).

### Camada Gold:

Na camada gold temos os dados já agregados oriundos da camada silver e, a partir daí, eles serão úteis para que possamos obter algumas respostas dos questionamentos feitos no início de nosso trabalho.

Uma estratégia interessante para esta parte do projeto (e que foi adotada neste processo) é a modelagem dimensional em formato estrela, bastante difundida na área de Engenharia de Dados.

O esquema estrela é particularmente estratégico no sentido de visar uma otimização de performance, simplicidade e ter maior eficiência nas análises de dados. Em outras palavras, podemos citar os seguintes benefícios de se usar o esquema estrela:

- Reduz a complexidade de consultas, uma vez que a tabela fato (em nosso projeto será a `gold_fact_covid_cases`) conecta-se diretamente às tabelas de dimensão (em nosso projeto serão a `gold_dim_country` e `gold_dim_date`) via chaves simples
- Podem ser feitas consultas SQL simples sem complexas normalizações
- Dimensões de tempo otimizadas, permitindo análises temporais sem reprocessar os dados brutos
- Separação clara entre dados e metadados, pois na tabela fato se encontram os dados numéricos (métricas quantitativas, ex: `total_cases`, `new_deaths`) e nas tabelas dimensão encontramos os contextos descritivos (`gold_dim_country`: países, PIB, etc e na `gold_dim_date`: datas)

Mediante estas vantagens, o esquema estrela mostra-se extremamente aderente ao contexto que queremos visualizar.

Evidenciamos, primeiramente, a criação da camada gold em nosso código.

```
dbutils.fs.mkdirs("dbfs:/mnt/covid_data/gold")
```

Figura 13 – Criação da camada gold

A seguir, também evidenciamos como foi feita a criação das tabelas fato e dimensão em nosso código:

```
# Tabela Fato:  
✓fact_df = spark.table("silver_covid_data").select(  
    "country_code",  
    "date",  
    "total_cases",  
    "new_cases",  
    "total_deaths",  
    "new_deaths"  
)  
  
fact_df.write.format("delta").mode("overwrite").save("/mnt/covid_data/gold/fact_covid_cases.delta")  
  
spark.sql("""  
CREATE TABLE IF NOT EXISTS gold_fact_covid_cases  
USING DELTA  
LOCATION '/mnt/covid_data/gold/fact_covid_cases.delta'  
""")
```

Figura 14 – Tabela Fato

```
# Dimensão País  
✓country_dim_df = spark.table("silver_covid_data").select(  
    "country_code",  
    "country_name",  
    "continent",  
    "population",  
    "gdp_per_capita",  
    "human_development_index"  
)  
.dropDuplicates(["country_code"])  
  
country_dim_df.write.format("delta").mode("overwrite").save("/mnt/covid_data/gold/dim_country.delta")  
  
spark.sql("""  
CREATE TABLE IF NOT EXISTS gold_dim_country  
USING DELTA  
LOCATION '/mnt/covid_data/gold/dim_country.delta'  
""")
```

Figura 15 – Tabela Dimensão País

```
# Dimensão Data
from pyspark.sql.functions import year, month, dayofmonth, dayofweek

date_dim_df = spark.table("silver_covid_data").select(
    "date"
).dropDuplicates().withColumn("year", year("date")) \
    .withColumn("month", month("date")) \
    .withColumn("day", dayofmonth("date")) \
    .withColumn("day_of_week", dayofweek("date"))

date_dim_df.write.format("delta").mode("overwrite").save("/mnt/covid_data/gold/dim_date.delta")

spark.sql("""
CREATE TABLE IF NOT EXISTS gold_dim_date
USING DELTA
LOCATION '/mnt/covid_data/gold/dim_date.delta'
""")
```

Figura 16 – Tabela Dimensão Data

Após a execução dos códigos de criação das tabelas, podemos observá-las que foram criadas no catálogo.

The screenshot shows the Databricks catalog interface. On the left, there's a sidebar with options like 'New', 'Workspace', 'Recents', 'Search', 'Catalog' (which is selected), 'Workflows', 'Compute', 'Machine Learning', and 'Experiments'. The main area is titled 'Data' and contains two sections: 'Databases' and 'Tables'. Under 'Databases', there's a dropdown menu set to 'default'. Under 'Tables', there's a dropdown menu set to 'Filter Tables'. A list of tables is shown, including 'bronze\_covid\_data', 'gold\_dim\_country', 'gold\_dim\_date', 'gold\_fact\_covid\_cases', and 'silver\_covid\_data'. There are also 'Create Table' and 'Delete' buttons at the top right of the main area.

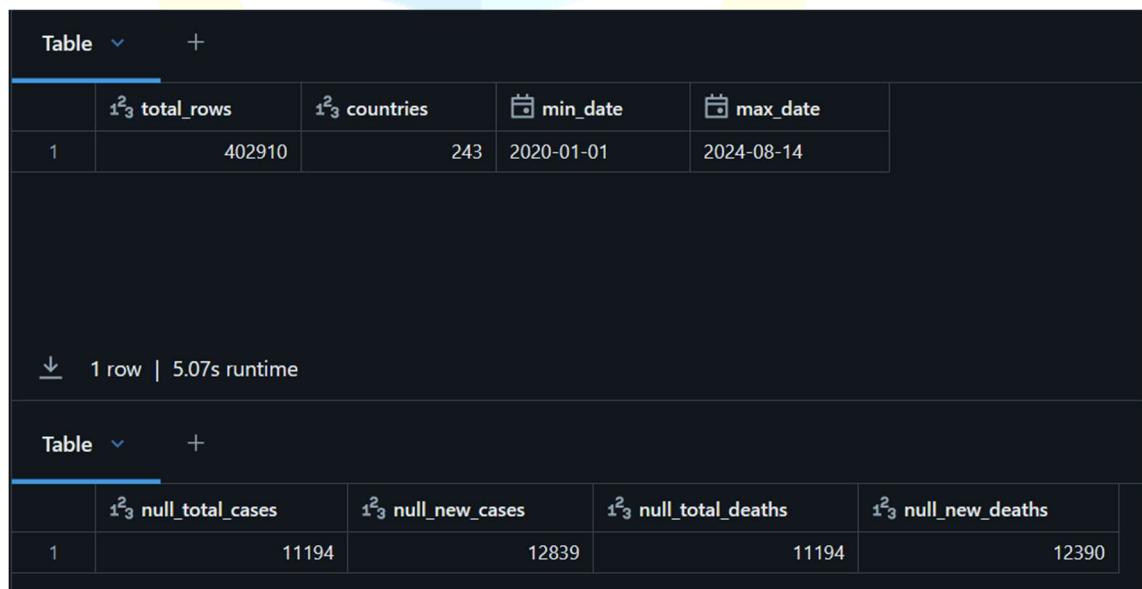
Figura 17 – Tabelas Gold incluídas no catálogo

E, da mesma forma que nas outras camadas, executamos também um processo de verificação de qualidade para verificar se tudo ocorreu a contento. Por exemplo, queremos ver a quantidade de linhas em que não tivemos casos nem de mortes.

```
# Verificação de qualidade dos dados da camada Gold:  
  
# Verificar completude dos dados  
display(spark.sql("""  
    SELECT  
        COUNT(*) as total_rows,  
        COUNT(DISTINCT country_code) as countries,  
        MIN(date) as min_date,  
        MAX(date) as max_date  
    FROM  
        gold_fact_covid_cases  
""")  
  
# Verificar valores nulos  
display(spark.sql("""  
    SELECT  
        COUNT(*) - COUNT(total_cases) as null_total_cases,  
        COUNT(*) - COUNT(new_cases) as null_new_cases,  
        COUNT(*) - COUNT(total_deaths) as null_total_deaths,  
        COUNT(*) - COUNT(new_deaths) as null_new_deaths  
    FROM  
        gold_fact_covid_cases  
""")  
""")
```

Figura 18 – Verificação de qualidade da camada gold

Como resultado, obtemos as tabelas de saída abaixo.



The screenshot shows two tables generated by the SQL queries in Figure 18. The first table has four columns: total\_rows, countries, min\_date, and max\_date. It contains one row with values 402910, 243, 2020-01-01, and 2024-08-14 respectively. The second table has five columns: null\_total\_cases, null\_new\_cases, null\_total\_deaths, and null\_new\_deaths. It also contains one row with values 11194, 12839, 11194, and 12390 respectively. Below each table, it says "1 row | 5.07s runtime".

	total_rows	countries	min_date	max_date
1	402910	243	2020-01-01	2024-08-14

	null_total_cases	null_new_cases	null_total_deaths	null_new_deaths
1	11194	12839	11194	12390

Figura 19 – Tabela de saída

Com o processo de criação das camadas executado com sucesso, estamos, a partir de agora, aptos a elaborar as métricas necessárias para respondermos aos questionamentos pelos quais nos propusemos a explicar.

## 6 ) Métricas:

Com o processo de ETL, podemos tentar responder ao propósito inicial do trabalho, com consultas SQL simples, as quais são mostradas logo adiante.

### Total de casos e mortes por continente:

Evidenciamos a seguir uma consulta SQL para tentar verificar o total de casos e mortes que aconteceram por continente.

```
SELECT
    c.continent,
    SUM(COALESCE(f.total_cases, 0)) AS total_cases,
    SUM(COALESCE(f.total_deaths, 0)) AS total_deaths,
    CASE
        WHEN SUM(COALESCE(f.total_cases, 0)) = 0 THEN 0
        ELSE ROUND((SUM(COALESCE(f.total_deaths, 0)) / SUM(COALESCE(f.total_cases, 0))) * 100, 2)
    END AS mortality_rate
FROM
    gold_fact_covid_cases f
JOIN
    gold_dim_country c ON f.country_code = c.country_code
GROUP BY
    c.continent
HAVING
    SUM(COALESCE(f.total_cases, 0)) > 0
ORDER BY
    total_cases DESC
```

Figura 20 – Consulta SQL para o total de casos e mortes

Vale salientar a importância do esquema estrela. Conforme dito anteriormente, o esquema ajuda no sentido de melhorar a performance nas consultas e assim foi feito na figura acima com a presença das tabelas fato e das de dimensão e com um filtro, com os dados já agrupados, utilizando a cláusula having em total\_cases. A saída podemos verificar logo abaixo na figura.

	A <sup>B</sup> C continent	1.2 total_cases	1.2 total_deaths	1.2 mortality_rate
1	Asia	252167317226	1790404880	0.71
2	Europe	236756684151	2361106039	1
3	North America	127073670231	1880636077	1.48
4	South America	73484570403	1646924372	2.24
5	Africa	14616886098	306955815	2.1
6	Oceania	11598053992	23933802	0.21

Figura 21 – Total de casos e mortes por continente

Um dado bônus muito interessante é a taxa de mortalidade que pode também ser utilizada para estudos e correlações futuras. Acredito que seja de grande valia para estabelecermos relações entre população e doença, bem como foi a profundidade da doença nos locais (relações de causa / efeito).

#### Evolução de casos em um determinado país:

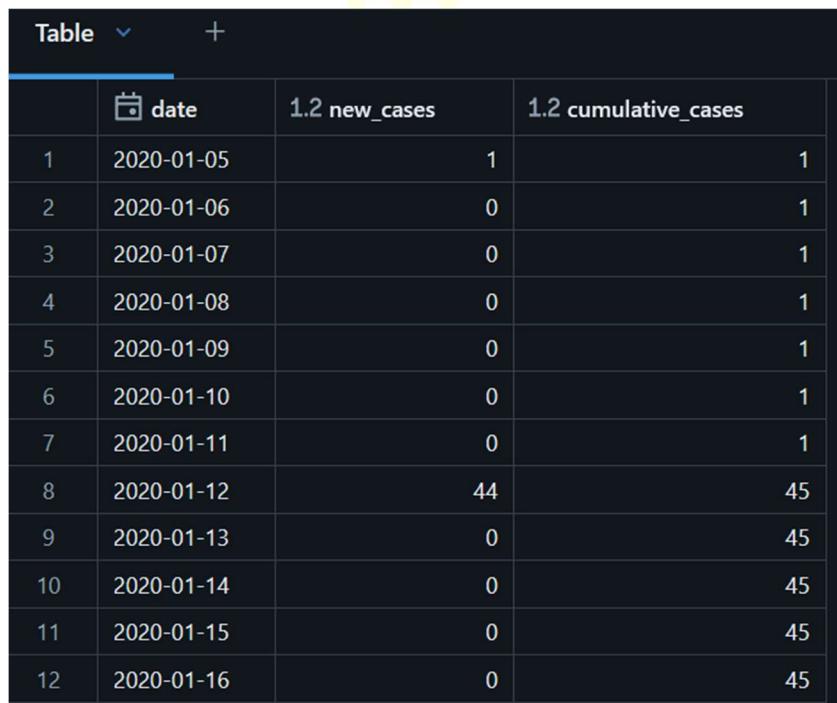
Outra métrica é a de evolução de casos pelos países, fazendo uma seleção a nossa escolha. Para efeito de exemplo, foi selecionada a China, uma vez que o local foi o marco zero da pandemia. Nada mais natural pensar como que essa evolução se deu, a princípio.

Evidenciamos, abaixo, a consulta SQL utilizada na demonstração.

```
SELECT
    d.date,
    f.new_cases,
    SUM(f.new_cases) OVER (ORDER BY d.date) as cumulative_cases
FROM
    gold_fact_covid_cases f
JOIN
    gold_dim_country c ON f.country_code = c.country_code
JOIN
    gold_dim_date d ON f.date = d.date
WHERE
    c.country_name = 'China'
ORDER BY
    d.date
```

Figura 22 – Consulta SQL para um determinado país (ex: China)

E, feita esta consulta, temos o resultado de acordo com os parâmetros utilizados. É importante observar que foi pensada uma evolução diária, haja vista a incidência da doença mesmo, onde no período inicial obtivemos vários dias com poucos ou nenhum caso e em outros com números astronômicos. A seguir são mostradas as primeiras linhas. A evolução fica mais clara quando, a partir daí, são criados gráficos ou outras formas mais amigáveis de visualização. Abordaremos isto mais à frente.



	date	1.2 new_cases	1.2 cumulative_cases
1	2020-01-05	1	1
2	2020-01-06	0	1
3	2020-01-07	0	1
4	2020-01-08	0	1
5	2020-01-09	0	1
6	2020-01-10	0	1
7	2020-01-11	0	1
8	2020-01-12	44	45
9	2020-01-13	0	45
10	2020-01-14	0	45
11	2020-01-15	0	45
12	2020-01-16	0	45

Figura 23 – Listagem dos casos novos e acumulados

#### Ranking Top-10 de países com maior taxa de mortalidade:

Outra métrica bastante interessante é um ranking com os países que tiveram maior taxa de mortalidade. Acredito ser útil pois, com estes dados, podemos verificar as possíveis falhas ou eventuais contextos destes países ao não obterem sucesso rápido no combate ao COVID. E, com isso, criar condições para poder executar melhorias suas políticas de forma ao problema não se repetir, em tese pelo menos.

Evidenciamos, a seguir, como foi elaborada a consulta SQL para este item.

```
SELECT
    c.country_name,
    MAX(f.total_cases) AS total_cases,
    MAX(f.total_deaths) AS total_deaths,
    MAX(f.total_deaths)/MAX(f.total_cases)*100 AS mortality_rate
FROM
    gold_fact_covid_cases f
JOIN
    gold_dim_country c ON f.country_code = c.country_code
WHERE
    f.total_cases > 1000
GROUP BY
    c.country_name
HAVING
    MAX(f.total_cases) > 10000
ORDER BY
    mortality_rate DESC
LIMIT 10
```

Figura 24 – Ranking Top 10 de países com maior taxa de mortalidade

E, com isso, obtemos a seguinte tabela de saída.



	country_name	total_cases	total_deaths	mortality_rate
1	Yemen	11945	2159	18.07450816241105
2	Sudan	63993	5046	7.885237447845858
3	Syria	57423	3163	5.508245824843703
4	Somalia	27334	1361	4.979146850076828
5	Peru	4526977	220975	4.881292747897769
6	Egypt	516023	24830	4.8118010243729445
7	Mexico	7619458	334551	4.390745378477052
8	Bosnia and Herzegovina	403666	16392	4.060782924496984
9	Afghanistan	235214	7998	3.400307804807537
10	Ecuador	1077445	36050	3.3458784439112903

Figura 25 – Tabela de saída Top 10

Note-se a ordenação decrescente da taxa de mortalidade, de forma a dar maior impacto sobre o assunto. E o resultado apresentado acredito que vá de encontro com o esperado, ou seja, a doença marcou mais nos países com política de combate mais deficitária, sobretudo a América Latina, trazendo um pouco mais para perto de nossa realidade. Também vale citar que nem sempre o país mais populoso teve a maior quantidade de mortes.

Com essas métricas, procurou-se responder àqueles questionamentos no início, que foram o cerne para o desenvolvimento do trabalho.

A título de melhorar relatórios, análises e apresentações para áreas de negócio e demais interessados, podemos elaborar dashboards através de visualizações gráficas, painéis ou até mesmo integrar os resultados dentro de plataformas robustas como, por exemplo, Power BI, Google Cloud, entre outras, de forma a facilitar futuras tomadas de decisão. Trataremos melhor sobre este assunto na sequência.

## **7 ) Dashboards:**

Os dashboards são uma forma de visualização das métricas resultantes de uma forma mais interativa e apresentável a fim de facilitar a compreensão das informações passadas.

Em nosso projeto, por exemplo, podemos citar os tópicos abaixo como benefícios da interatividade:

- ❖ Identificar desproporções regionais na disseminação do vírus: Uma vez que podemos encontrar algum padrão geopolítico (tipo Europa com mais casos do que a África ou, então, inferir capacidades de testagem (países com maior infraestrutura médica tendem a reportar mais casos)
- ❖ Efetividade das políticas de contenção ao longo do tempo: Identificar ondas de infecção relacionadas a relaxamento de medidas, além de verificar tendências no comportamento da doença
- ❖ Falhas na condução da doença: Podem ser inferidas alguma informação sobre qualidade do sistema de saúde, subnotificação de casos ou, até mesmo, indicadores socioeconômicos

Estes são só alguns dos benefícios. O que vale destacar é o poder desta ferramenta, pois contribui para termos diagnósticos cada vez mais precisos e melhorar constantemente a gestão no tocante às tomadas de decisão.

Evidenciamos, a seguir, os dashboards utilizados para as métricas deste projeto.

#### Relação de mortes e casos por continente:

Fizemos uma pequena variação, onde mostramos o total de casos por continente, a distribuição de casos por continente e a taxa de mortalidade por continente. Evidenciamos, a seguir, o gráfico resultante através do Plotly.<sup>3</sup>

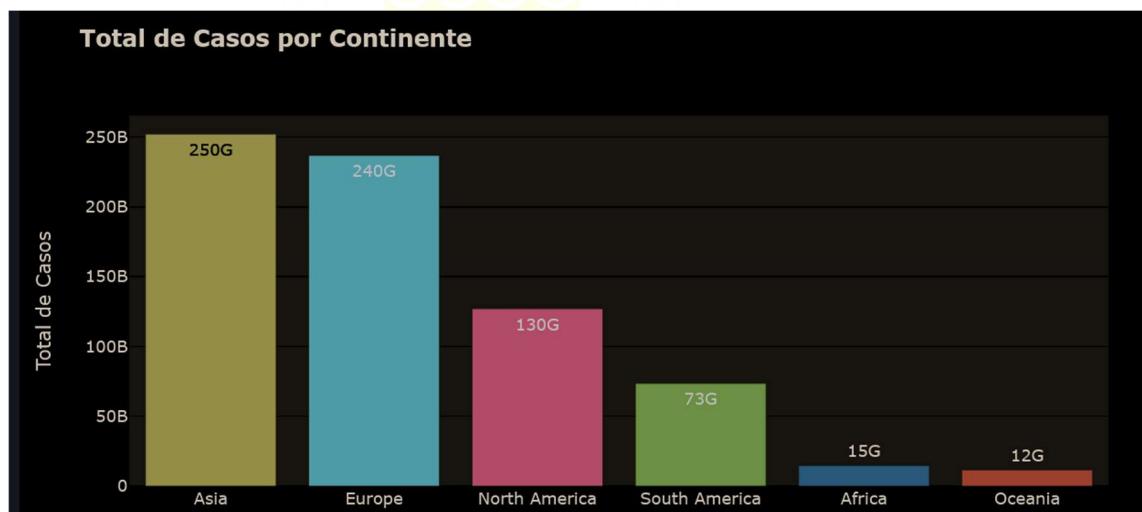


Figura 26 – Dashboard sobre total de casos por continente

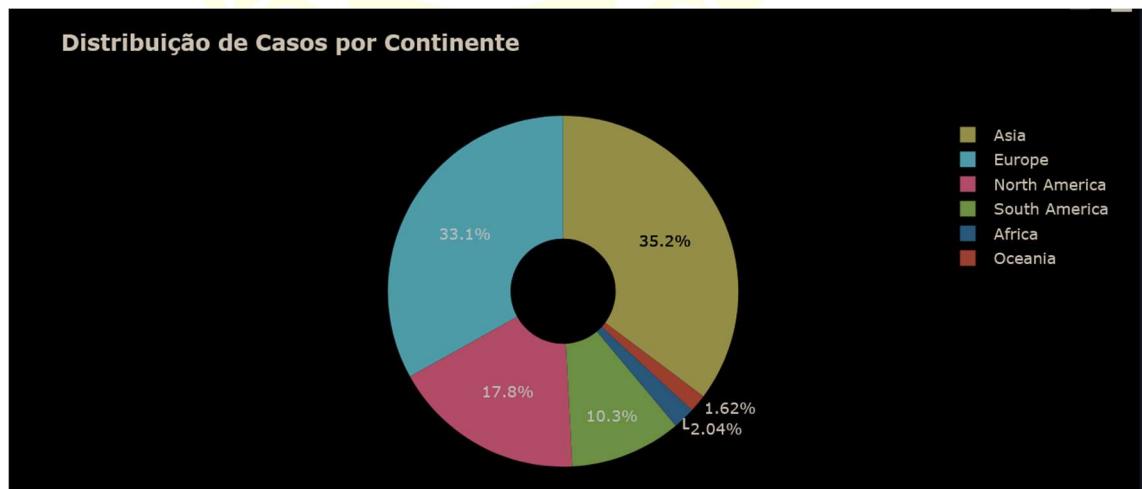


Figura 27 – Dashboard de Distribuição de casos por continente

<sup>3</sup> O código completo de construção do gráfico está disponível no repositório do Github ([https://github.com/rodrigobsouza17/Engenharia\\_de\\_Dados/tree/main/MVP](https://github.com/rodrigobsouza17/Engenharia_de_Dados/tree/main/MVP)).

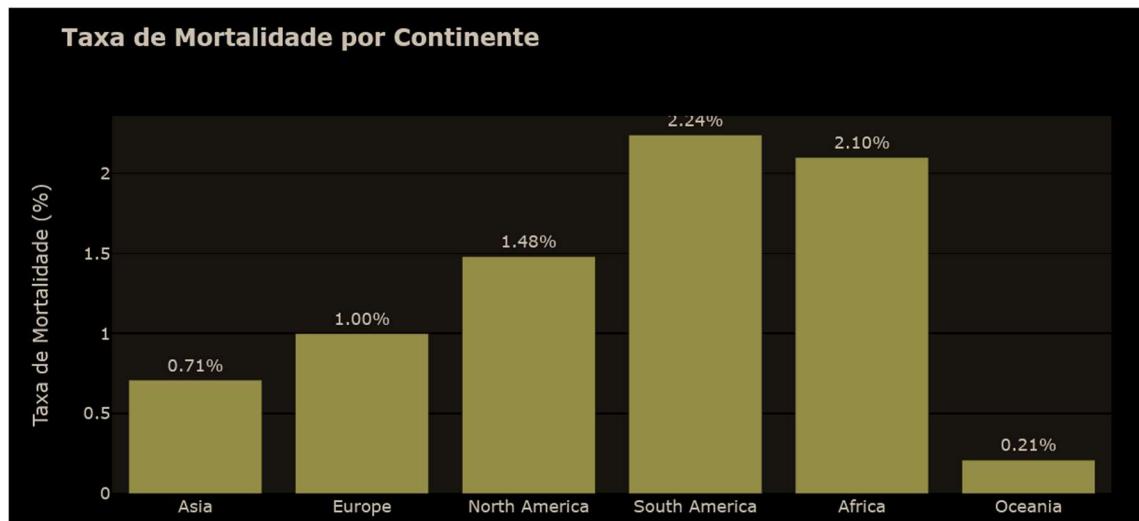


Figura 28 – Dashboard sobre taxa de mortalidade por continente

São boas visualizações onde verificamos muitos casos na Europa e Ásia, porém existiram também muitas mortes na África e Américas (Norte e Sul), cujos dados corroboram a falha que houve nestas regiões na condução do problema.

#### Evolução de casos em um determinado país:<sup>4</sup>

Evidenciamos, neste caso, sob a forma de gráfico, para evidenciar a evolução temporal da doença. Infelizmente, a escala não foi a ideal, porém a ideia pensada acho válida para fins didáticos, sobretudo em nosso projeto.

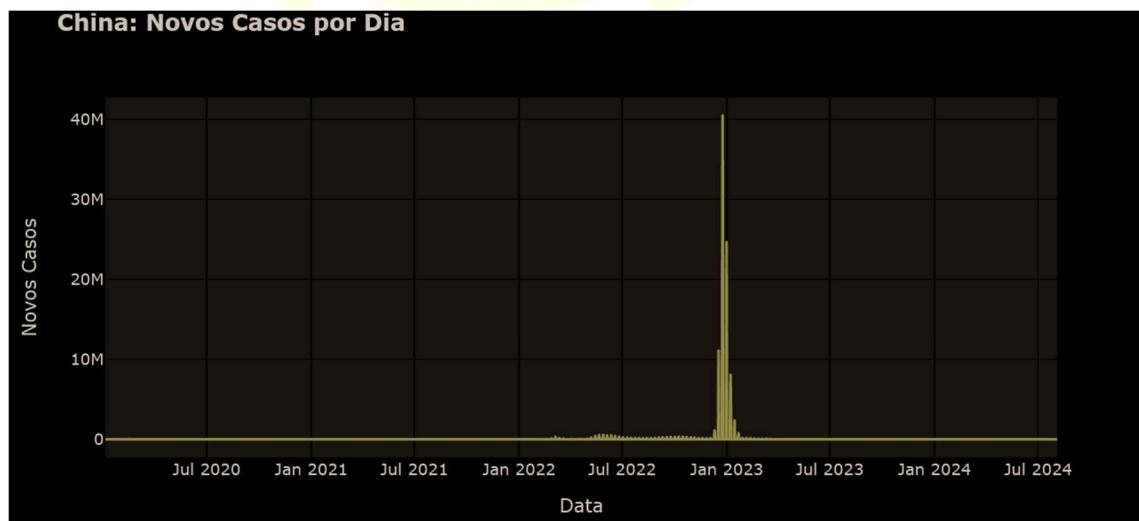


Figura 29 – Dashboard sobre evolução em um país (ex: China)

<sup>4</sup> O código completo de construção do gráfico está disponível no repositório do Github ([https://github.com/rodrigobsouza17/Engenharia\\_de\\_Dados/tree/main/MVP](https://github.com/rodrigobsouza17/Engenharia_de_Dados/tree/main/MVP)).



Figura 30 – Dashboard sobre casos acumulados (ex: China)

Com esses dashboards podemos notar, sobretudo na descoberta da nova variante, um aumento expressivo de casos, demandando grande atenção na época. Podemos inferir também que, mesmo com um aumento expressivo, a taxa de mortalidade caiu, o que demonstra um entendimento do cenário e melhoria de como se proceder com a doença, exemplificada na figura abaixo.



Figura 31 – Dashboard gráfico da taxa de mortalidade

Ranking top-10 dos países com maior taxa de mortalidade<sup>5</sup>:

Optou-se, neste caso, por um gráfico de barras, que dá uma boa noção de quantificação dos casos, além de uma separação em cores pelos continentes, ficando uma visualização bem satisfatória e compreensível da informação transmitida.

O gráfico resultante é visualizado logo abaixo.

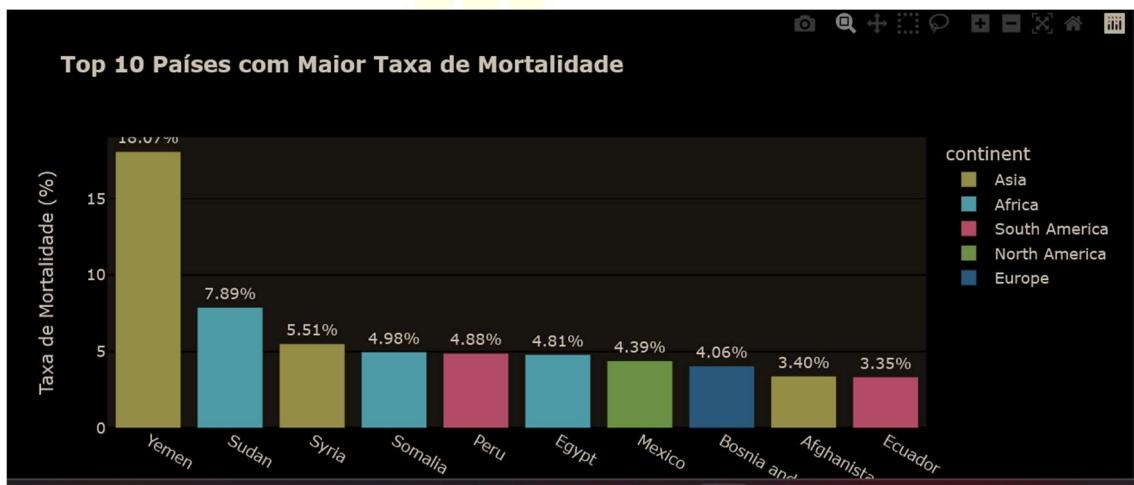


Figura 32 – Ranking Top 10 dos países com maior taxa de mortalidade

O resultado apresentado surpreendeu, em parte, em relação ao que se achava, pois realmente alguns países que tiveram grandes dificuldades no combate à doença estão listados (ex: Peru, Equador) mas a disparidade no primeiro lugar foi bem grande e por um país que, sinceramente, não me lembro de estar cotado nas listagens na época. Agora, por ser da Ásia (exatamente Oriente Médio), certamente passou pelos problemas que vivenciamos de perto.

<sup>5</sup> O código completo de construção do gráfico está disponível no repositório do Github ([https://github.com/rodrigobsouza17/Engenharia\\_de\\_Dados/tree/main/MVP](https://github.com/rodrigobsouza17/Engenharia_de_Dados/tree/main/MVP)).

## **8 ) Autoavaliação:**

Este projeto foi altamente desafiador, por se tratar de uma disciplina totalmente nova e não ter absolutamente nenhuma experiência nesta parte de Engenharia de Dados. E depois por não ter no início, digamos, grandes ideias de projetos. Mas sempre tive a curiosidade de estudar o tema da pandemia e, em especial, analisar a COVID, daí acabou sendo o ponto de partida. Portanto, acredito que foi muito produtivo o trabalho, a fim de acrescentar conhecimento e ter mais intimidade com as operações em plataformas de nuvem. Por exemplo, a Databricks era um mundo inteiramente desconhecido. Achei interessante efetuar todo o processo nela, mesmo com as limitações existentes na versão *Community Edition*, mas que não me atrapalharam em nada. Em suma, para fins didáticos, a experiência foi excelente. Também tinha pouca experiência no uso do SQL. Possuo boa vivência com o *python*, porém ao operar em conjunto os dois módulos eu já não tinha o suporte necessário para fazer alguma coisa. E o projeto foi bastante enriquecedor nesta parte. Obviamente, ainda há muito ainda para se aprender, contudo a experiência me ajudou muito para não só adquirir mais conhecimento como também plantar aquela sementinha de continuar buscando mais informações e mais soluções para incrementá-las na vida profissional.

Em resumo, a experiência foi plenamente válida e, com certeza, replicarei os conhecimentos para disseminar a informação e me desenvolver cada vez mais.

Aproveito também o espaço para agradecer aos professores do curso pela oportunidade de apresentação do assunto. Contribuiu bastante para o meu aprendizado.

## 9 ) Anexos:

### Catálogo de Dados:

Este documento descreve todos os conjuntos de dados utilizados no pipeline, desde a camada de ingestão (bronze) até a camada analítica (gold), garantindo rastreabilidade, governança e clareza tanto para usuários técnicos quanto para usuários não-técnicos.

### Metadados das camadas:

Camada Bronze	
Nome da tabela	bronze_covid_data
Fonte	Our World in Data (OWID)
Descrição	Dados brutos globais de casos, mortes e indicadores socioeconômicos
Formato	Delta Lake
Frequência de Atualização	Diária
Localização	dbfs:/mnt/covid_data/bronze/owid-covid-data.delta
Campos chave	iso_code: Código do país location: Nome do país date: Data do registro total_cases: Casos acumulados new_cases: Novos casos no dia

Camada Silver	
Nome da tabela	silver_covid_data
Descrição	Dados limpos e padronizados, com tratamentos de qualidade
Transformações aplicadas	Remoção de registros sem continente Conversão de datas para formato DATE Renomeação de colunas (ex: de new_cases para novos_casos)
Regras de qualidade	total_cases: não pode ser negativo country_code: não pode ser nulo

Camada Gold		
Tabela Fato		
	Nome	gold_fact_covid_cases
	Chaves	(country_code, date)
	Métricas	total_cases: Casos acumulados new_cases: Novos casos no dia mortality_rate: $(\text{total_deaths}/\text{total_cases}) * 100$
Tabela Dimensão		
	Nome	gold_dim_country
	Chave	country_code
	Atributos	continent: Região geográfica gdp_per_capita: Indicador econômico population: População total
Tabela Dimensão		
	Nome	gold_dim_date
	Chave	date
	Atributos	year, month, day : Hierarquia temporal

### Dicionário de Dados:

Campo	Tipo	Descrição	Exemplo
country_code	STRING	Código ISO do país	"BRA" (Brasil)
total_cases	LONG	Casos confirmados até a data	35000000
mortality_rate	DOUBLE	% de mortes em relação a casos	2.8 (2.8%)
gdp_per_capita	DOUBLE	PIB per capita em USD	14500.75
date	DATE	Data do registro (YYYY-MM-DD)	2023-05-15

### Fluxo de Dados:

Basicamente, o projeto foi estruturado para seguir o esquema representado na imagem abaixo:

```
A[Fonte OWID] -->|CSV| B[Camada Bronze]  
B -->|Limpeza| C[Camada Silver]  
C -->|Modelagem| D[Gold: Fato + Dimensões]  
D --> E[Dashboards]
```

Figura 33 – Fluxo de Dados

Relacionamentos:

Os relacionamentos são do tipo esquema estrela, ou seja, relacionam-se com dimensões.

A figura abaixo traduz como o processo foi conduzido para este trabalho.

```
gold_fact_covid_cases ||--o{ gold_dim_country : "country_code (N:1)"  
gold_fact_covid_cases ||--o{ gold_dim_date : "date (N:1)"
```

Figura 34 – Relacionamentos

Ou seja, são relações de cardinalidade muitos para um em que há uma relação entre duas tabelas onde uma coluna pode ter vários valores e na outra tabela tem apenas um valor para cada.

Procuramos apresentar com este esquemático simples um resumo bem claro de todas as etapas do trabalho para facilitar o entendimento.

### Criação dos clusters na Databricks:

Uma das dificuldades que obtive ao dar os primeiros passos na plataforma Databricks foi o conceito de cluster. Entretanto, consegui entender e desenvolver o trabalho com a ajuda da documentação do sistema mais o suporte (principalmente!) dado pelos colegas de classe. E coloco logo abaixo as evidências de como foi o processo de criação dos clusters pois sem eles o notebook não funciona.

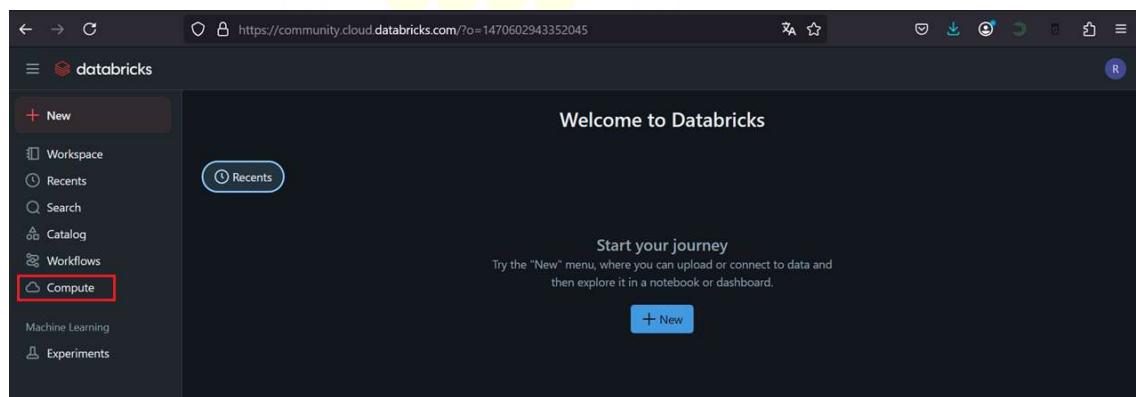


Figura 35 – Criação do Cluster na Databricks

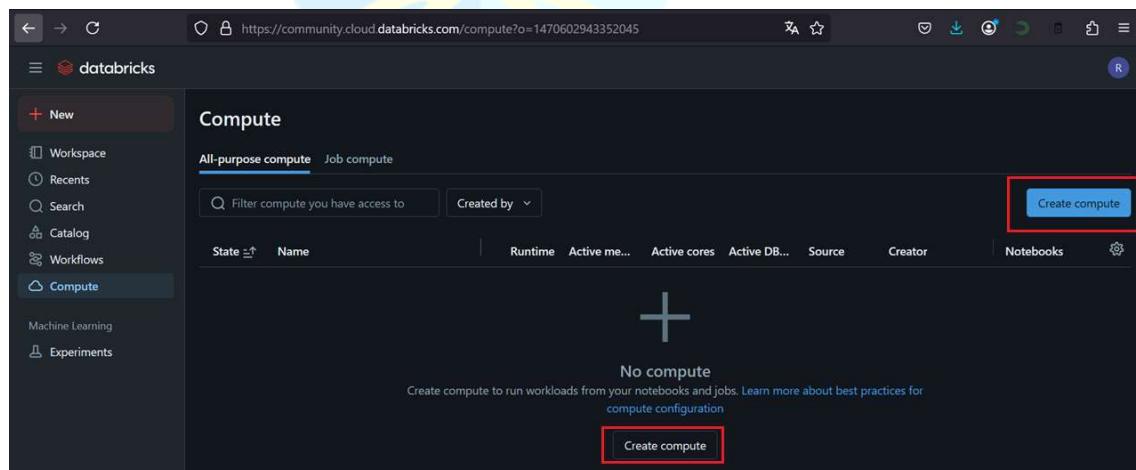


Figura 36 – Criação do Cluster na Databricks

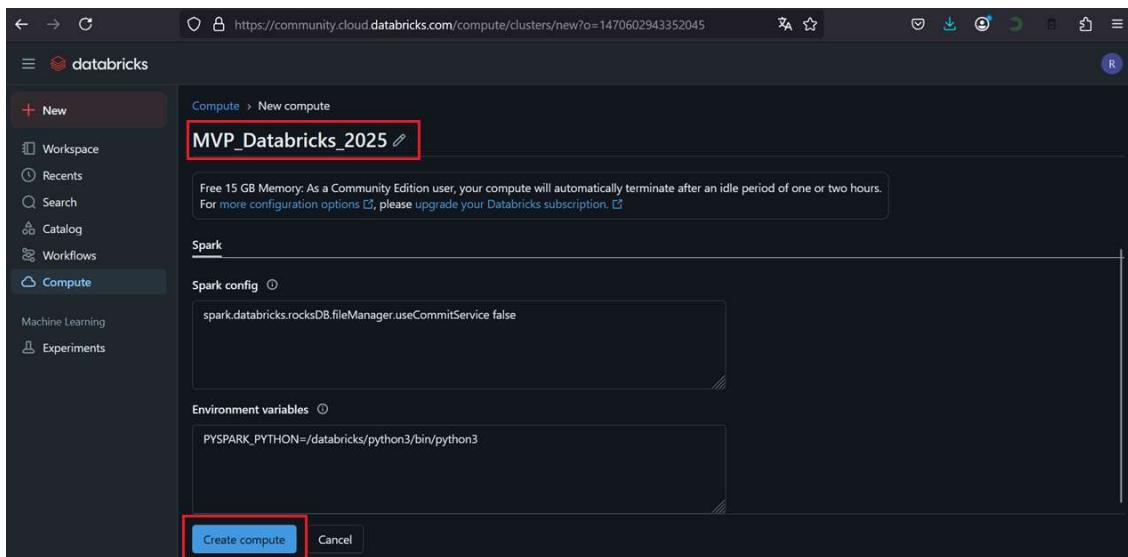


Figura 37 – Criação do Cluster na Databricks

Após isto, o cluster é criado e podemos começar as nossas configurações para o projeto (Para estar pronto, fica uma bolinha verde ao lado do nome do cluster)

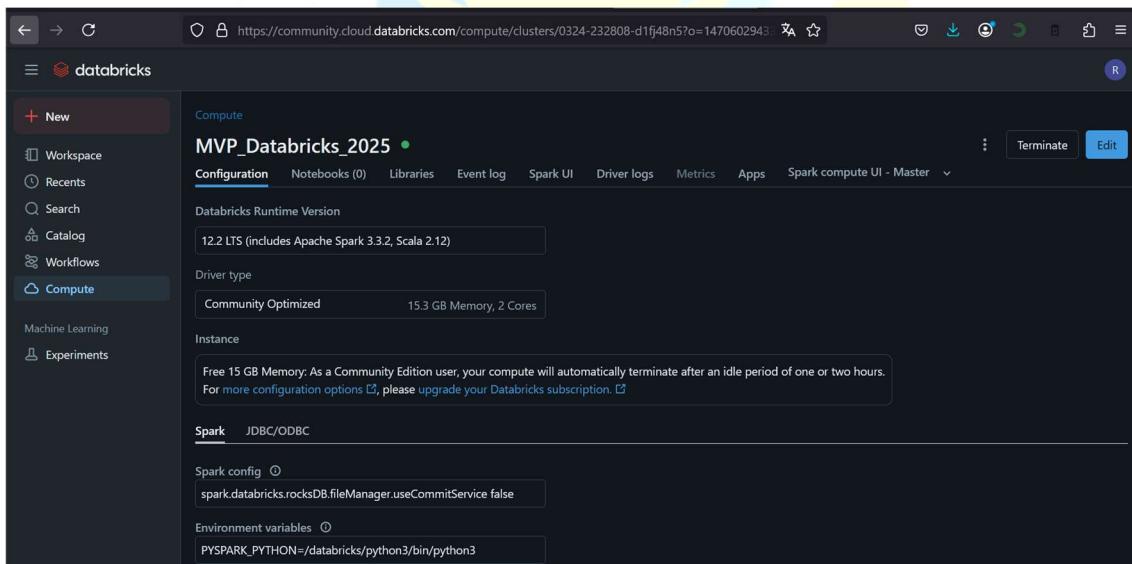


Figura 38 – Criação do Cluster na Databricks

OBS: A plataforma Databricks Community Edition possui uma limitação no quesito dos clusters, pois se ficarem inativos por uma hora, a plataforma desliga os e não conseguimos mais utilizar o cluster criado. Para contornar este problema, devemos:

1 ) Ir em Workspace (Espaço de trabalho):

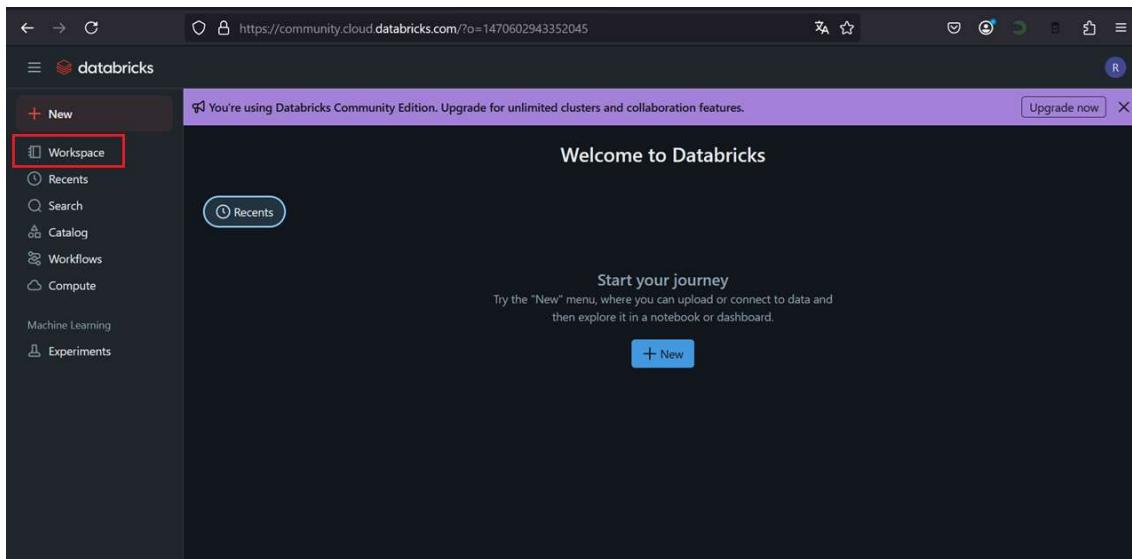


Figura 39 – Criação do Cluster na Databricks

2 ) Selecionar o Notebook a ser trabalhado

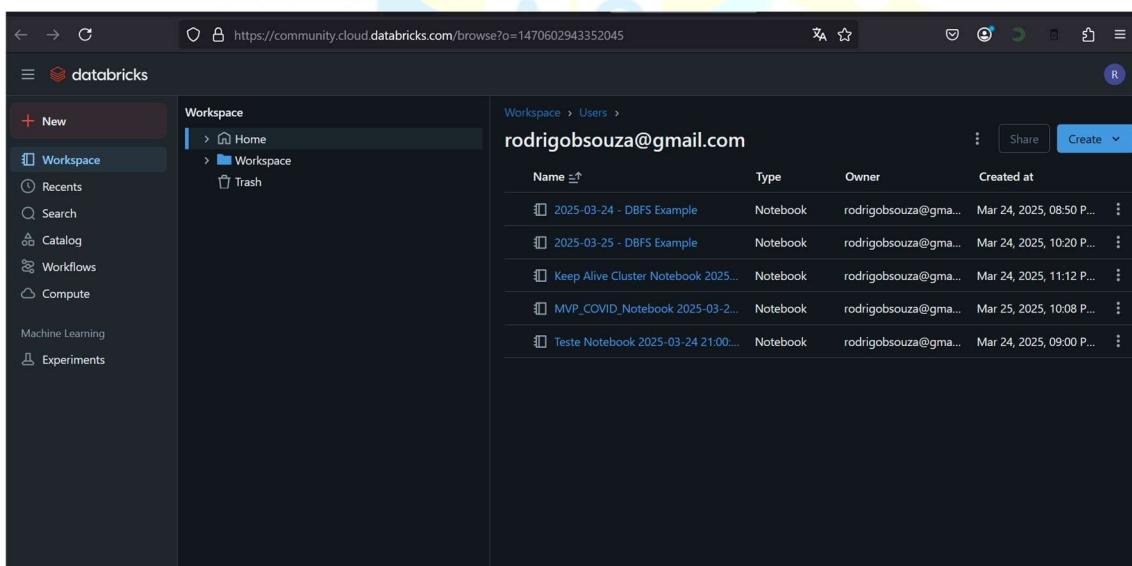
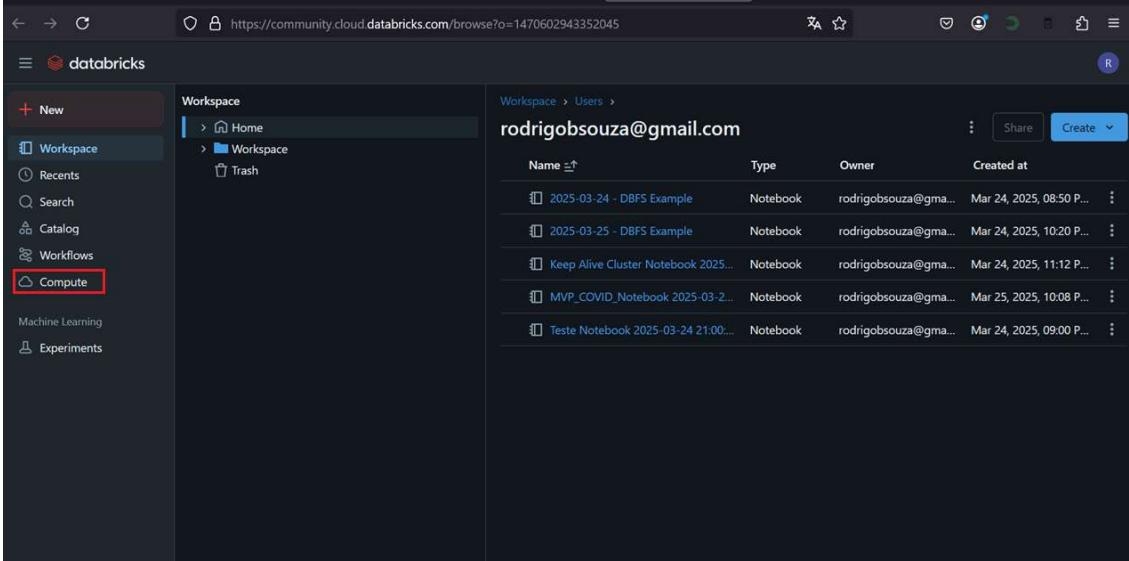


Figura 40 – Criação do Cluster na Databricks

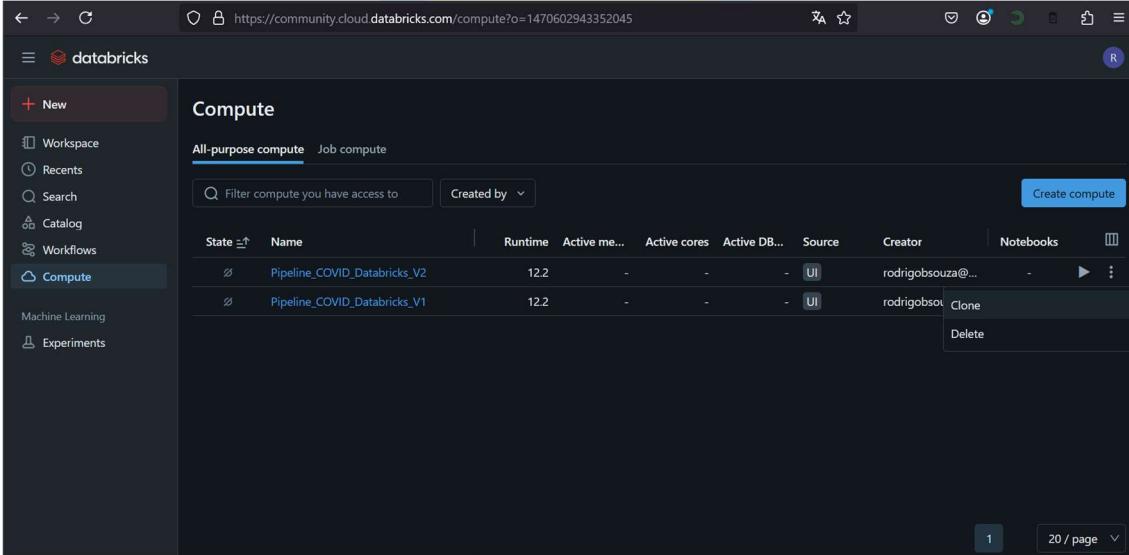
3 ) Para facilitar, clicar em Compute abrindo em outra guia



The screenshot shows the Databricks web interface. The left sidebar has a dark theme with various navigation options: '+ New', 'Workspace' (selected), 'Recents', 'Search', 'Catalog', 'Workflows', 'Compute' (highlighted with a red box), 'Machine Learning', and 'Experiments'. The main content area shows the 'Workspace > Users > rodrigobsouza@gmail.com' view. It lists several notebooks created by the user, including '2025-03-24 - DBFS Example', '2025-03-25 - DBFS Example', 'Keep Alive Cluster Notebook 2025...', 'MVP\_COVID\_Notebook 2025-03-2...', and 'Teste Notebook 2025-03-24 21:00...'. A 'Share' and 'Create' button are visible at the top right of the list.

Figura 41 – Criação do Cluster na Databricks

4 ) Ao usar mais de uma vez, vão estar lá gravados os clusters anteriores utilizados. Para criar um novo, pode selecionar qualquers deles, clicar nos 3 pontinhos verticais e depois em Clone



The screenshot shows the 'Compute' page in the Databricks web interface. The left sidebar is identical to Figure 41. The main content area is titled 'Compute' and shows two entries under 'All-purpose compute': 'Pipeline\_COVID\_Databricks\_V2' and 'Pipeline\_COVID\_Databricks\_V1'. Both entries have a 'Runtime' of 12.2, 'Active me...' and 'Active cores' of -, and a 'Source' of 'UI'. The 'Creator' is 'rodrigobsouza@gmail.com' and the 'Notebooks' column shows '-' for both. For the second entry ('V1'), there is a context menu open with options 'Clone' and 'Delete'. At the bottom right, there are pagination controls showing page 1 of 20.

Figura 42 – Criação do Cluster na Databricks

5 ) Digitar o nome do novo Cluster a ser criado e clicar em Create compute

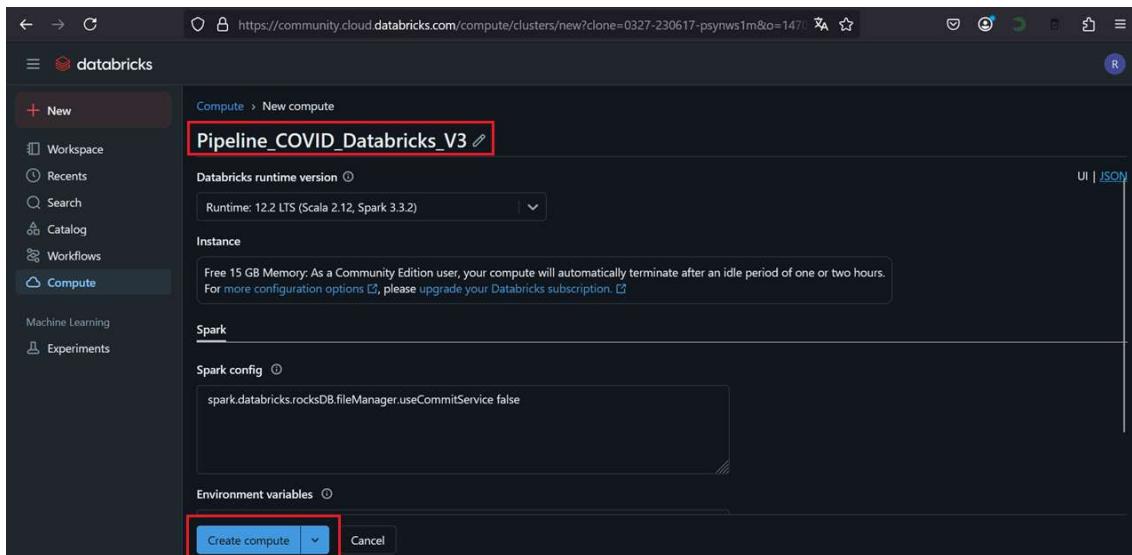


Figura 43 – Criação do Cluster na Databricks

6 ) Ir no Notebook e selecionar o cluster que acabou de ser criado



Figura 44 – Criação do Cluster na Databricks

E, pronto! A partir daí estamos prontos para executar todas as células do Notebook.