

Análise de Dados da COVID-19 em Minas Gerais: Estruturação de dados e Visualização de Informações

**Daniela Moreira da Silva, Luan Lacerda Ramos, Rafael Silva de Alcântara, Victor
Pinheiro Louvisi, Wander de Carvalho, Alisson Rabelo Arantes**

PUC Minas

Curso de Tecnologia em Banco de Dados

`danimoreirasv@gmail.com, luan_ramosl@hotmail.com,
rafael.dtnrd@gmail.com, victorlouvisi@yahoo.com.br,
wanderdecarvalho@gmail.com, alissonr@sga.pucminas.br`

Resumo: *A pandemia causada pelo vírus SARS-CoV-2 se disseminou rapidamente pelo mundo, impactando profundamente as populações, economias e sistemas de saúde dos países afetados. Minas Gerais, um dos maiores estados do Brasil, continua a enfrentar desafios significativos relacionados à pandemia, tornando-se crucial uma análise dos dados coletados nos últimos anos. Viu-se necessário, portanto, estruturar uma base de dados com registros consolidados dos casos de COVID-19 no estado de Minas Gerais no período entre 2020 e 2021. Os objetivos que se pretende alcançar com a estruturação da referida base de dados são: a) descrição das bases de dados utilizadas, b) elaboração da arquitetura a ser construída para o tratamento dos dados, c) descrição detalhada da metodologia a ser utilizada no tratamento dos dados, d) inserção dos dados obtidos em algum SGBD relacional, e) limpeza e transformação dos dados após sua estruturação no SGBD e f) análise dos dados e demonstração dos resultados obtidos através da construção de gráficos. A criação de um banco de dados relacional tornará a base mais performática em termos de uso e de manipulação, melhorando de forma significativa a interação dos seus usuários com os dados e permitindo a análise dos dados a fim de responder questões relevantes. Como resultados, se constatou a existência preliminar de uso de dois métodos para obtenção e tratamento de dados, seja ele através do uso da linguagem Python, juntamente com SQL, ou através do assistente de importação do SSMS. Além dessa observação, no que tange à visualização de dados, o uso do Power BI se mostrou eficaz e eficiente para que os usuários finais possam interagir, visualizar e obter insights sobre o tema abordado.*

1. Introdução

A pandemia de COVID-19, causada pelo vírus SARS-CoV-2, emergiu no final de 2019 e se disseminou rapidamente pelo mundo, desencadeando uma crise global de saúde pública e impactando profundamente as populações, economias e sistemas de saúde dos países afetados. Quase cinco anos depois, apesar dos esforços para conter a disseminação do vírus e vacinar a população, Minas Gerais, um dos maiores estados do Brasil em termos territoriais, continua a enfrentar desafios significativos relacionados à pandemia.

Em vista deste cenário, torna-se crucial uma análise dos dados coletados nos últimos anos, para compreender o cenário atual e decidir qual a melhor forma de direcionar os recursos do Estado, buscando a melhora da qualidade de vida da população.

A partir desta problemática, viu-se necessária a estruturação de uma base de dados com os registros consolidados dos casos de COVID-19, no estado de Minas Gerais, entre o período de 2020 e 2021, tido como o auge da pandemia no Brasil. Essa consolidação deve ser de fácil acesso e permitir a fácil observação dos dados, mitigando os esforços necessários à obtenção dos mesmos, oferecendo uma base estruturada e que permita a realização das análises pertinentes.

Desta forma, os objetivos que se pretende alcançar com a modelagem da referida base de dados são: a) descrição das bases de dados utilizadas, b) elaboração da arquitetura a ser construída para o tratamento dos dados, c) descrição detalhada da metodologia a ser utilizada no tratamento dos dados, d) inserção dos dados obtidos em algum SGBD relacional, e) limpeza e transformação dos dados após sua estruturação no SGBD e f) análise dos dados e construção de gráficos a fim de gerar conhecimento a respeito da evolução dos casos da COVID-19 no Estado de Minas Gerais e seu impacto na sua população. Ou seja, busca-se saber quais foram as regiões e perfis demográficos mais afetados pela doença, relativo ao número de atendimentos, internações e óbitos.

Assim, tem-se justificada a criação e manipulação da base de dados pretendida pelo presente trabalho, visto a grande relevância de gerar conhecimentos a respeito dos impactos causados pela COVID-19 no Estado de Minas Gerais através do tratamento e da análise dos dados coletados pelas instituições do mesmo.

2. Descrição das bases de dados utilizadas

A base de dados utilizada contém registros sobre os casos de COVID-19, no Estado de Minas Gerais, entre os anos de 2020 e 2021. Estes dados são de domínio público, com licença de *Creative Commons Attribution*, tendo sido compilados pela Secretaria de Estado da Saúde de Minas Gerais.

A base de dados foi acessada no dia 08 de setembro de 2023, às 15:49, pelo portal de dados abertos do Governo Federal (<https://dados.gov.br/dados/conjuntos-dados/casos-confirmados-Covid-19>), e encontra-se no formato XLSX. A base de dados possui 752.500 linhas com observações referentes a casos confirmados de Covid-19 no Estado de Minas Gerais, e sua estrutura está de acordo com as descrições na tabela a seguir:

Tabela 1. Estrutura da base de dados utilizada na íntegra.

Nome da coluna/campo	Descrição	Tipo
ID	Código exclusivo do exame realizado	Inteiro
URS	Unidade Regional de Saúde em que o exame foi realizado	Caractere
MICRO	Microrregião em que o exame foi realizado	Caractere
MACRO	Macrorregião em que o exame foi realizado	Caractere
DATA_NOTIFICACAO	Data de notificação do resultado do exame realizado	Data
CLASSIFICACAO_CASO	Resultado do exame realizado	Caractere
SEXO	Sexo do paciente (Masculino ou Feminino)	Caractere
IDADE	Idade do paciente	Inteiro
FAIXA_ETARIA	Faixa etária do paciente	Caractere
MUNICIPIO_RESIDENCIA	Município de residência do paciente.	Caractere
CODIGO	Código do IBGE relativo ao município de residência do paciente	Inteiro
EVOLUCAO	Evolução do quadro do paciente	Caractere
DATA_EVOLUCAO	Data da evolução do quadro do paciente	Data
DATA_1_SINTOMA	Data em que o paciente apresentou os primeiros sintomas	Data
INTERNACAO	Dado referente a situação de internação em leito clínico do paciente	Caractere
UTI	Dado referente a situação de internação em leito UTI do paciente	Caractere

RACA	Raça declarada pelo paciente (Branca, Preta, Amarela, Parda ou Indígena)	Caractere
ETNIA	Nomes das etnias indígenas. Nome e código da etnia do paciente, quando indígena	Caractere
COMORBIDADE	Dado referente à presença ou ausência de comorbidade do paciente	Caractere
DATA_ATUALIZACAO	Dado referente a data de atualização, por parte da equipe da Sala de Situação, do resultado do paciente nos sistemas do Laboratório.	Data
ORIGEM_DA_INFORMACAO	Dado referente ao Sistema de origem da informação	Caractere

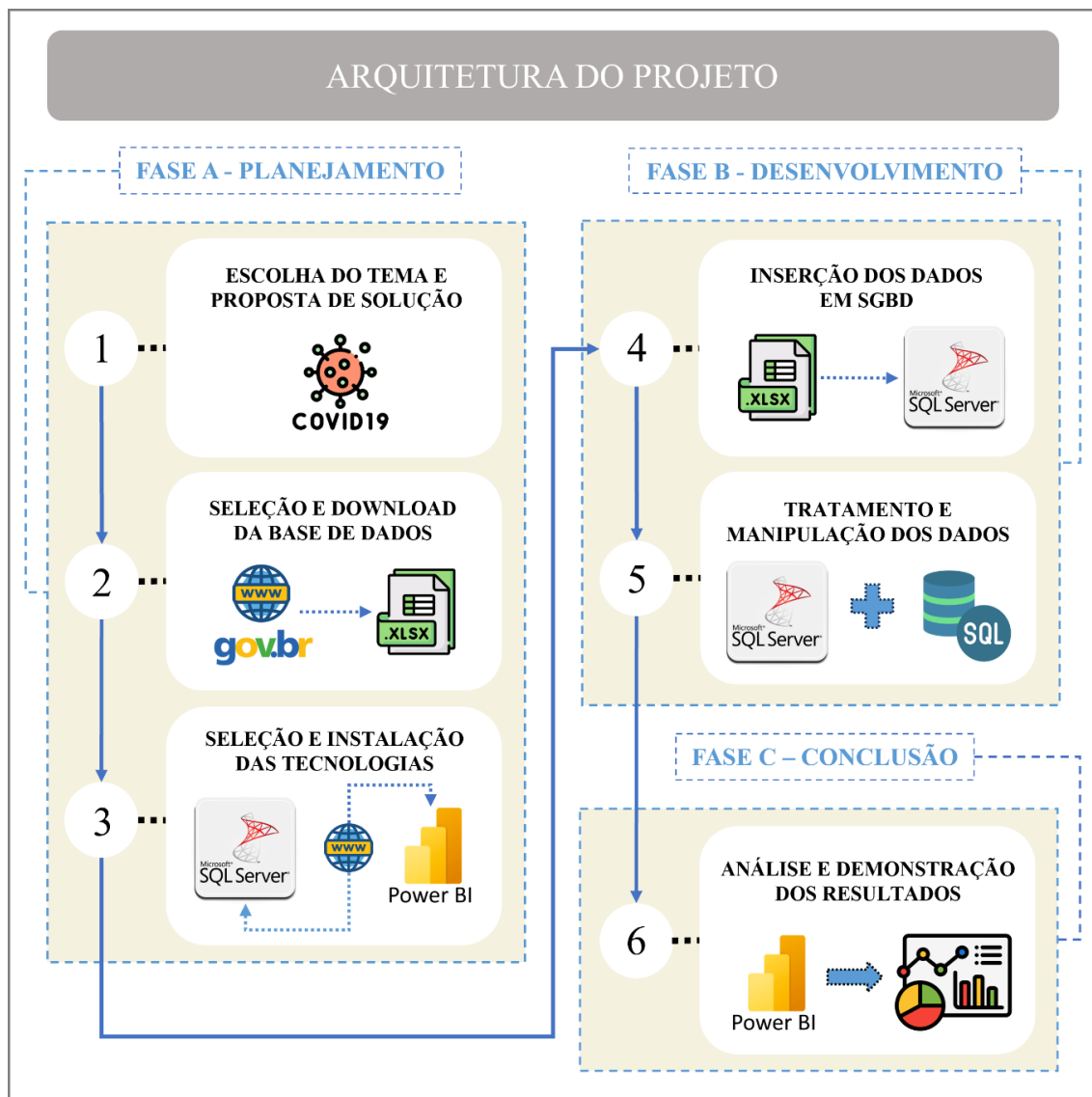
Fonte: Elaborado pelos autores (2023).

As primeiras colunas da tabela são referentes ao campo de chave de identificação do registro e a dados relativos à região de atendimento do paciente. As colunas seguintes são de data da notificação e resultado do exame de Covid-19, além de dados demográficos e de saúde de cada paciente, como sexo, idade, faixa etária, raça, município de residência, existência ou não de comorbidades e se o paciente passou por processo de internação ou não, seja ela comum ou em UTI. Finalmente, as colunas ‘DATA_ATUALIZACAO’ e ‘ORIGEM_DA_INFORMACAO’ são metadados que informam sobre a data em que os dados foram registrados e o Sistema onde foram registrados.

3. Arquitetura

A arquitetura do projeto foi descrita graficamente em um diagrama, o qual está representado na Figura 1, a seguir:

Figura 1. Diagrama de Arquitetura do Projeto



Fonte: Elaborado pelos autores (2023).

4. Metodologia

A metodologia do projeto foi dividida em três fases: planejamento, desenvolvimento e conclusão, os quais, como um todo, somam um total de 6 passos que serão realizados seguidamente, a fim de atingir os objetivos propostos na introdução deste trabalho.

Na fase A, de planejamento, são realizadas as atividades de planejamento, que incluem os passos um a três. O primeiro passo foi a escolha do tema do presente trabalho, que é análise de dados dos casos de COVID-19 do Estado de Minas Gerais.

O segundo passo da fase A da metodologia é onde se busca obter os dados de casos COVID-19 do Estado de Minas Gerais de uma fonte confiável, que, neste caso, se trata do portal dados.gov.br, que contém dados abertos do Governo Federal, Estados e Municípios do Brasil. Os dados foram extraídos desta fonte no formato de XLSX.

No terceiro passo da fase de planejamento, que corresponde ao passo de seleção e instalação das tecnologias que serão utilizadas, foram escolhidos o SQL Server, da Microsoft, como SGBD e o Power Bi para desenvolvimento de dashboards, uma vez que ambas são ferramentas robustas e amplamente utilizadas atualmente.

A fase B da metodologia do projeto, que engloba os passos quatro e cinco, corresponde à fase de desenvolvimento do projeto.

Uma vez obtidos os dados, no passo quatro, estes serão importados para um banco no SQL Server, onde poderão ser manipulados e gerenciados no passo cinco da metodologia, que envolve o processo de ETL (do inglês, *'Extract, Transform and Load'* – Extrair, Transformar e Carregar), o qual busca garantir que os dados sejam consistentes e estejam prontos para análise. Neste passo, faremos a importação utilizando um script SQL gerado por um programa ou o Assistente de Importação e Exportação do SQL Server, e então manipularemos os dados com o uso de comandos da linguagem SQL (que é utilizada para 'comunicar-se' com o banco de dados), para corrigir valores, excluir colunas desnecessárias, etc.

Com os dados da COVID-19 tratados, a próxima etapa é a análise visual usando o Power BI. O Power BI permite criar painéis interativos e relatórios dinâmicos para acompanhar a evolução da pandemia. Ele se conecta diretamente ao SQL Server para buscar os dados tratados e exibi-los em gráficos, mapas e tabelas interativas.

Finalmente, na fase de conclusão da metodologia, que engloba o passo seis, serão respondidas a perguntas propostas na introdução deste trabalho, com o uso da linguagem DAX (*Data Analysis Expressions*) e/ou SQL. A linguagem DAX no Power BI permite realizar cálculos personalizados, como taxas de crescimento, análises comparativas entre regiões e previsões com base nos dados da pandemia. Além disso, será gerado um dashboard com os gráficos e tabelas mais relevantes.

Em resumo, esta metodologia visa oferecer um processo estruturado para análise de dados dos casos de COVID-19 do Estado de Minas Gerais, desde a aquisição de dados,

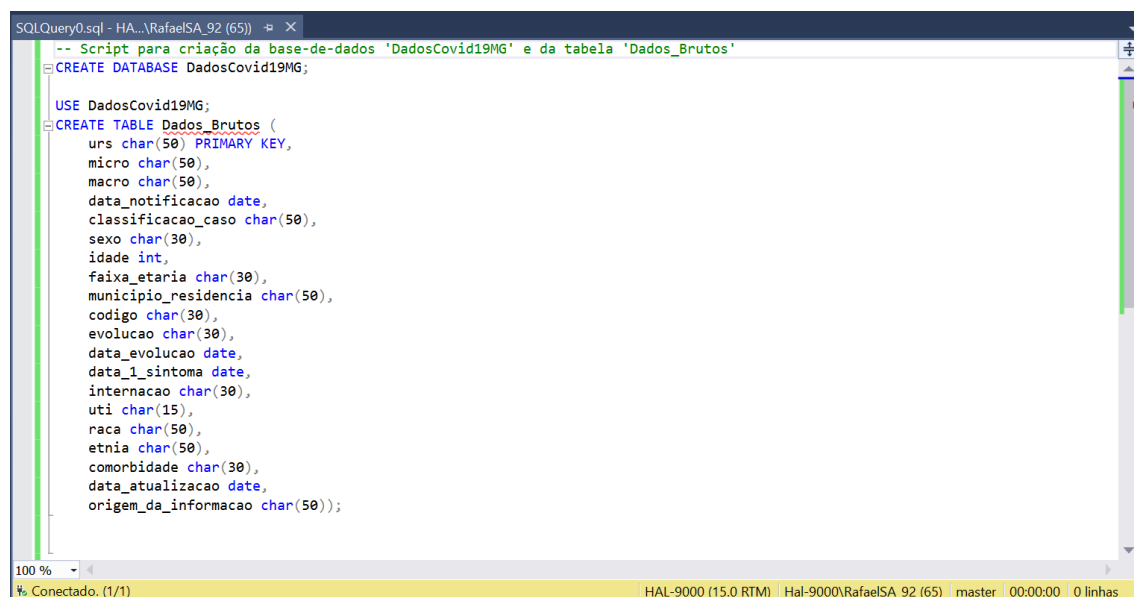
importação para o SQL Server, processo de ETL, análise visual no Power BI, até a resposta a perguntas críticas com a linguagem DAX e/ou SQL, tendo em vista a importância de registrar precisamente todos os passos que foram realizados para garantir a qualidade e confiabilidade do processo.

5. Captura dos dados

Após baixar o arquivo ‘.xlsx’ com os dados da fonte informada, os dados precisaram ser importados para um SGBD para que ficassem armazenados com segurança e para que fossem tratados nas fases posteriores. O SGBD escolhido para o projeto foi o SQL Server, em virtude de sua robustez e qualidade.

Para importação dos dados para o SGBD escolhido, primeiramente foi necessário criar um banco-de-dados e uma tabela utilizando-se os comandos ‘CREATE DATABASE’ e ‘CREATE TABLE’ da linguagem SQL, que é a linguagem utilizada para interagir com bancos-de-dados. Abaixo, na Figura 2, tem-se o script SQL utilizado para criação do banco-de-dados e da tabela.

Figura 2. Script SQL para criação da base-de-dados e tabela



```
-- Script para criação da base-de-dados 'DadosCovid19MG' e da tabela 'Dados_Brutos'
CREATE DATABASE DadosCovid19MG;

USE DadosCovid19MG;
CREATE TABLE Dados_Brutos (
    urs char(50) PRIMARY KEY,
    micro char(50),
    macro char(50),
    data_notificacao date,
    classificacao_caso char(50),
    sexo char(30),
    idade int,
    faixa_etaria char(30),
    municipio_residencia char(50),
    codigo char(30),
    evolucao char(30),
    data_evolucao date,
    data_1_sintoma date,
    internacao char(30),
    uti char(15),
    raca char(50),
    etnia char(50),
    comorbidade char(30),
    data_atualizacao date,
    origem_da_informacao char(50));
```

Fonte: Elaborado pelos autores (2023).

Após criação do banco-de-dados e da tabela no SGBG, há dois métodos principais para fazer a importação dos dados: geração de script(s) SQL por meio de um programa ou via assistente de importação do SQL server. Ambas as formas foram testadas para saber qual seria a mais eficiente.

5.1. Importação via script SQL gerado por programa

Na primeira forma de importação dos dados (importação via script SQL), foi desenvolvido um pequeno programa utilizando-se a linguagem Python, que é uma

linguagem de programação de alto nível, interpretada de script, e com o auxílio da plataforma de computação interativa baseada na web Jupyter Notebook.

A finalidade do programa desenvolvido foi gerar um comando ‘INSERT’ do SQL para cada uma das observações contidas na tabela de dados original e escrevê-los em um arquivo ‘.sql’ o qual seria posteriormente importado e lido pelo SGBD, inserindo, desta forma, os dados na tabela do banco-de-dados no SGBD criada previamente.

A seguir, são mostrados os trechos código do programa desenvolvido, descrevendo-se o que cada um realiza.

No primeiro trecho do código, são importadas as bibliotecas necessárias para o desenvolvimento do programa, que foram a biblioteca ‘csv’, para leitura de arquivos .csv, a biblioteca ‘pandas’, para manipulação de ‘data-frames’, e a biblioteca ‘re’, de expressões regulares. No segundo trecho, o arquivo ‘.xlsx’ baixado do site gov.br é lido, utilizando-se o método ‘pd.read_excel()’, da biblioteca pandas, e os dados são alocados na variável ‘df’. Já no terceiro e quatro trechos, os valores ausentes são preenchidos com o valor 0 e os valores da coluna ‘IDADE’ são convertidos para o tipo ‘int32’ (essa conversão se fez necessária para que os valores numéricos não fossem escritos com aspas no script gerado). Finalmente, no trecho cinco, os dados transformados são exportados em um arquivo ‘.csv’ nomeado ‘Dados_Brutos.csv’. Os trechos mencionados estão ilustrados pela Figura 3, abaixo:

Figura 3. Trechos um a cinco do código do programa de geração de script SQL

```
# Importação das bibliotecas necessárias
import csv
import pandas as pd
import re

# Importação do arquivo Excel
df = pd.read_excel('Dados_Brutos.xlsx')

# Preenche os valores ausentes com 0
df.fillna(value=0, inplace=True)

# Converte os valores da coluna 'IDADE' para int32
df['IDADE'] = df['IDADE'].astype(int)

# Transfere os dados convertidos para um arquivo .csv
df.to_csv('Dados_Brutos.csv', index=False)
```

Fonte: Elaborado pelos autores (2023).

Após a exportação do arquivo ‘Dados_Brutos.csv’, foi criada uma função/método chamada ‘scriptGenerator’, que recebe três parâmetros: ‘nome_arquivo’ (nome do arquivo de script que será gerado, que deve terminar com ‘.sql’), ‘limite_inferior’ (define qual a primeira linha que deve ser escrita no arquivo) e ‘limite_superior’ (define a última linha que deve ser escrita no arquivo).

Este método abre o arquivo ‘Dados_Brutos.csv’ no modo leitura, cria um arquivo com o nome definido como argumento, abre o arquivo no modo escrita e então itera cada

uma das linhas do arquivo ‘.csv’, removendo apóstrofos (palavras que contém apóstrofos causam erro de execução, visto que apóstrofos são lidos como aspas simples, que, por sua vez, são usadas para inserir valores do tipo caractere e data pelo SQL Server), trocando os valores 0 por ‘NULL’ (que define um valor ausente no SQL Server), colocando as palavras e datas entre parênteses, e finalmente escrevendo os comandos ‘INSERT’ para cada uma das linhas no arquivo ‘.sql’.

A seguir, na Figura 4, é mostrado este trecho do código:

Figura 4. Trecho seis do código do programa de geração de script SQL

```
# Função para gerar script de SQL
def scriptGenerator(nome_arquivo, limite_inferior, limite_superior):
    # Abre o arquivo 'Dados_Brutos.csv' no modo leitura
    with open('Dados_Brutos.csv', newline='') as csvfile:
        reader = csv.reader(csvfile)

        # Cria um arquivo '.sql' com o parâmetro recebido e abre o arquivo no modo de escrita
        f = open(nome_arquivo, 'w')

        counter = 0

        for line in reader:
            newline = list()
            # Remove todas as apóstrofes das palavras
            for element in line:
                if re.search("'", element):
                    index = element.find("'")
                    e_list = list(element)
                    e_list[index] = ''
                    delimiter = ''
                    element = delimiter.join(e_list)
                    element = "'" + element + "'"
                    newline.append(element)
                # Substitui todos os elementos 0 por 'NULL'
                elif element == '0':
                    element = 'NULL'
                    newline.append(element)
                # Coloca todas as palavras e datas entre aspas simples
                elif not (element.isdigit() or element == 'NULL'):
                    element = "'" + element + "'"
                    newline.append(element)
                else:
                    newline.append(element)

            # Une os elementos de cada linha, separados por vírgula
            delimiter = ','
            row = delimiter.join(newline)

            # Cria os comandos insert para cada linha da tabela e escreve em um arquivo '.sql'
            if row.startswith('\ID\'): continue
            if counter >= limite_inferior and counter < limite_superior:
                f.write('INSERT INTO Dados_Brutos VALUES (' + row + ')\n\n')

            counter += 1

        f.close()
```

Fonte: Elaborado pelos autores (2023).

Finalmente, no último trecho do código, a função/método ‘*scriptGenerator*’ é chamada oito vezes para gerar oito scripts de SQL com 100.000 comandos ‘INSERT’ em cada um. Foi necessário gerar oito scripts porque, ao tentar fazer a inserção dos dados utilizando-se apenas um script com 752.500 comandos ‘INSERT’, o SGBD gerava um erro e, após tentar inserir os dados com scripts de vários tamanhos, percebeu-se que o SQL server só conseguia executar 100.000 comandos ‘INSERT’ por vez.

Abaixo, nas Figuras 5 e 6, respectivamente, são mostrados o trecho do código com as chamadas da função e um trecho do script gerado:

Figura 5. Trecho final do código do programa de geração de script SQL

```
# Gera scripts SQL com um número de linhas consecutivas pré-definido nos argumentos da função
scriptGenerator('script1.sql',0,100000)
scriptGenerator('script2.sql',100000,200000)
scriptGenerator('script3.sql',200000,300000)
scriptGenerator('script4.sql',300000,400000)
scriptGenerator('script5.sql',400000,500000)
scriptGenerator('script6.sql',500000,600000)
scriptGenerator('script7.sql',600000,700000)
scriptGenerator('script8.sql',700000,800000)
```

Fonte: Elaborado pelos autores (2023).

Figura 6. Trecho do arquivo gerado pelo programa de geração de scripts SQL

```
script1.sql - HAL-9...0\RafaelSA_92 (51) | SQLQuery0.sql - HA...RafaelSA_92 (65)
INSERT INTO Dados_Brutos VALUES (1, 'TEOFILO OTONI', 'TEOFILO OTONI/MALACACHETA', 'NORDESTE', '14/06/2020', 'CASO CONFIRMADO', 'MASCULINO', 51, '50 A 59 ANOS', 'TEOFILO OTONI', '316860.0', 'EM ACOMPANHAMENTO', '09/06/2020', NULL, 'SIM', 'NAO', 'NAO INFORMADO', NULL, 'NAO INFORMADO', '23/06/21', 'SIVEP')
INSERT INTO Dados_Brutos VALUES (2, 'BELO HORIZONTE', 'BELO HORIZONTE/NOVA LIMA/CAETE', 'CENTRO', '24/06/2020', 'CASO CONFIRMADO', 'MASCULINO', 37, '30 A 39 ANOS', 'BELO HORIZONTE', '310620.0', 'EM ACOMPANHAMENTO', NULL, NULL, 'NAO INFORMADO', 'NAO INFORMADO', 'NAO INFORMADO', NULL, 'NAO INFORMADO', '23/06/21', 'SIVEP')
INSERT INTO Dados_Brutos VALUES (3, 'SAO JOAO DEL REI', 'SAO JOAO DEL REI', 'CENTRO SUL', '21/06/2020', 'CASO CONFIRMADO', 'MASCULINO', 77, '70 A 79 ANOS', 'LAGOA DO ANIL', '313740.0', 'EM ACOMPANHAMENTO', NULL, NULL, 'SIM', 'NAO INFORMADO', 'NAO INFORMADO', NULL, 'NAO INFORMADO', '23/06/21', 'SIVEP')
INSERT INTO Dados_Brutos VALUES (4, 'SAO JOAO DEL REI', 'SAO JOAO DEL REI', 'CENTRO SUL', '21/06/2020', 'CASO CONFIRMADO', 'MASCULINO', 60, '60 A 69 ANOS', 'CORONEL FABRICIANO', '311970.0', 'EM ACOMPANHAMENTO', NULL, NULL, 'SIM', 'NAO INFORMADO', 'NAO INFORMADO', NULL, 'SIM', '23/06/21', 'SIVEP')
INSERT INTO Dados_Brutos VALUES (5, 'ITUIUTABA', 'ITUIUTABA', 'TRIANGULO DO NORTE', '20/05/2020', 'CASO CONFIRMADO', 'FEMININO', 39, '30 A 39 ANOS', 'ITUIUTABA', '313420.0', 'RECUPERADO', '21/05/2020', NULL, 'SIM', 'NAO', 'NAO INFORMADO', NULL, 'SIM', '23/06/21', 'SIVEP')
INSERT INTO Dados_Brutos VALUES (6, 'CORONEL FABRICIANO', 'IPATINGA', 'VALE DO ACO', '13/06/2020', 'CASO CONFIRMADO', 'FEMININO', 35, '30 A 39 ANOS', 'JOANESIA', '313610.0', 'EM ACOMPANHAMENTO', NULL, NULL, 'NAO', 'NAO', 'NAO INFORMADO', NULL, 'SIM', '23/06/21', 'SIVEP')
INSERT INTO Dados_Brutos VALUES (7, 'DIAMANTINA', 'DIAMANTINA', 'JEQUITINHONHA', '19/06/2020', 'CASO CONFIRMADO', 'MASCULINO', 72, '70 A 79 ANOS', 'DIAMANTINA', '312160.0', 'EM ACOMPANHAMENTO', NULL, NULL, 'SIM', 'NAO', 'NAO INFORMADO', NULL, 'SIM', '23/06/21', 'SIVEP')
INSERT INTO Dados_Brutos VALUES (8, 'GOVERNADOR VALADARES', 'GOVERNADOR VALADARES', 'LESTE', '12/06/2020', 'CASO CONFIRMADO', 'MASCULINO', 39, '30 A 39 ANOS', 'GOVERNADOR VALADARES', '312770.0', 'EM ACOMPANHAMENTO', NULL, NULL, 'SIM', 'NAO', 'NAO INFORMADO', NULL, 'NAO', '23/06/21', 'SIVEP')
INSERT INTO Dados_Brutos VALUES (9, 'UBA', 'UBA', 'SUDESTE', '20/05/2020', 'CASO CONFIRMADO', 'MASCULINO', 75, '70 A 79 ANOS', 'UBA', '05/06/2020', NULL, 'NAO INFORMADO', 'NAO INFORMADO', 'BRANCA', NULL, 'NAO INFORMADO', '23/06/21', 'SIVEP')
INSERT INTO Dados_Brutos VALUES (10, 'UBERLANDIA', 'UBERLANDIA/ARAGUARI', 'TRIANGULO DO NORTE', '14/06/2020', 'CASO CONFIRMADO', 'MASCULINO', 56, '50 A 59 ANOS', 'UBERLANDIA', '317020.0', 'EM ACOMPANHAMENTO', NULL, NULL, 'SIM', 'NAO', 'PARDA', NULL, 'NAO INFORMADO', '23/06/21', 'SIVEP')
INSERT INTO Dados_Brutos VALUES (11, 'BELO HORIZONTE', 'CONTAGEM', 'CENTRO', '14/06/2020', 'CASO CONFIRMADO', 'FEMININO', 50, '50 A 59 ANOS', 'CONTAGEM', '311860.0', 'EM ACOMPANHAMENTO', NULL, NULL, 'SIM', 'NAO', 'PARDA', NULL, 'NAO INFORMADO', '23/06/21', 'SIVEP')
INSERT INTO Dados_Brutos VALUES (12, 'UBERLANDIA', 'UBERLANDIA/ARAGUARI', 'TRIANGULO DO NORTE', '14/06/2020', 'CASO CONFIRMADO', 'MASCULINO', 46, '40 A 49 ANOS', 'UBERLANDIA', '317020.0', 'EM ACOMPANHAMENTO', NULL, NULL, 'SIM', 'NAO', 'PARDA', NULL, 'NAO INFORMADO', '23/06/21', 'SIVEP')
INSERT INTO Dados_Brutos VALUES (13, 'TEOFILO OTONI', 'TEOFILO OTONI/MALACACHETA', 'NORDESTE', '14/06/2020', 'CASO CONFIRMADO', 'MASCULINO', 63, '60 A 69 ANOS', 'TEOFILO OTONI', '316860.0', 'RECUPERADO', NULL, NULL, 'SIM', 'NAO', 'BRANCA', NULL, 'NAO INFORMADO', '23/06/21', 'SIVEP')
INSERT INTO Dados_Brutos VALUES (14, 'BELO HORIZONTE', 'CONTAGEM', 'CENTRO', '14/06/2020', 'CASO CONFIRMADO', 'MASCULINO', 8, '1 A 9 ANOS', 'CONTAGEM', '311860.0', 'RECUPERADO', NULL, NULL, 'SIM', 'NAO', 'AMARELA', NULL, 'NAO INFORMADO', '23/06/21', 'SIVEP')
```

Fonte: Elaborado pelos autores (2023).

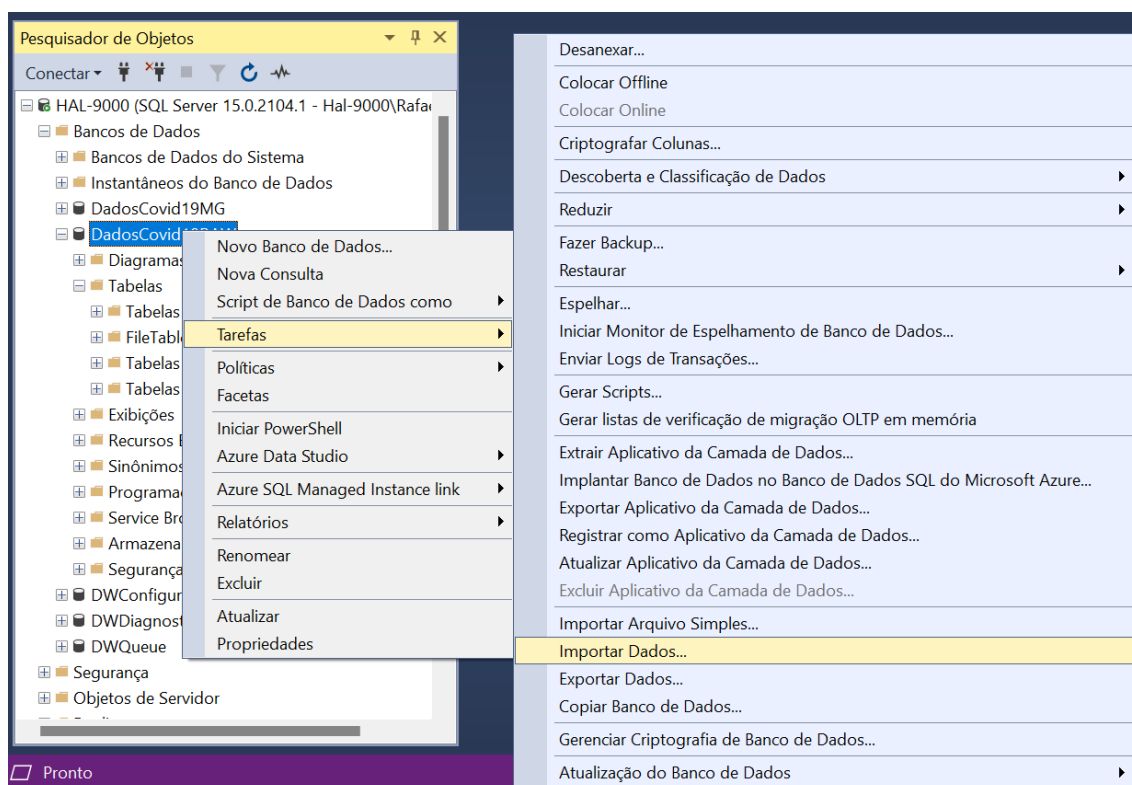
5.2. Importação via Assistente de Importação e Exportação do SQL Server

Assim como na importação via programa, para importar os dados utilizando o Assistente de Importação e Exportação do SQL Server (AISS), primeiramente é necessário criar um banco-de-dados, utilizando-se o comando ‘CREATE DATABASE’, como mostrado anteriormente. Após a criação do banco-de-dados, pode-se proceder com a importação de

dados do Excel para o Microsoft SQL Server Management Studio (SSMS) com auxílio do AIESS, sendo este um processo que acontece nos bastidores, envolvendo várias etapas.

Primeiro, é necessário abrir o AIESS, e isso pode ser feito clicando-se com o botão direito em cima da base-de-dados criada e selecionando-se ‘Tarefas’ em depois ‘Importar Dados’, como ilustrado pela Figura 7. Talvez seja necessário baixar e instalar alguma versão do Microsoft Access Engine, o qual instala componentes que permitem a transferência de arquivos do Excel para o SQL Server. Além disso, caso a arquitetura da máquina utilizada seja de 64bits, será necessário instalar a versão 64bits do assistente de importação e abri-la por meio do Menu Iniciar, visto que a versão padrão aberta pelo SSMS é a de 32bits, que pode gerar erros de compatibilidade com o Microsoft Access Engine 64bits instalado.

Figura 7. Caminho para abrir o AIESS dentro do SSMS

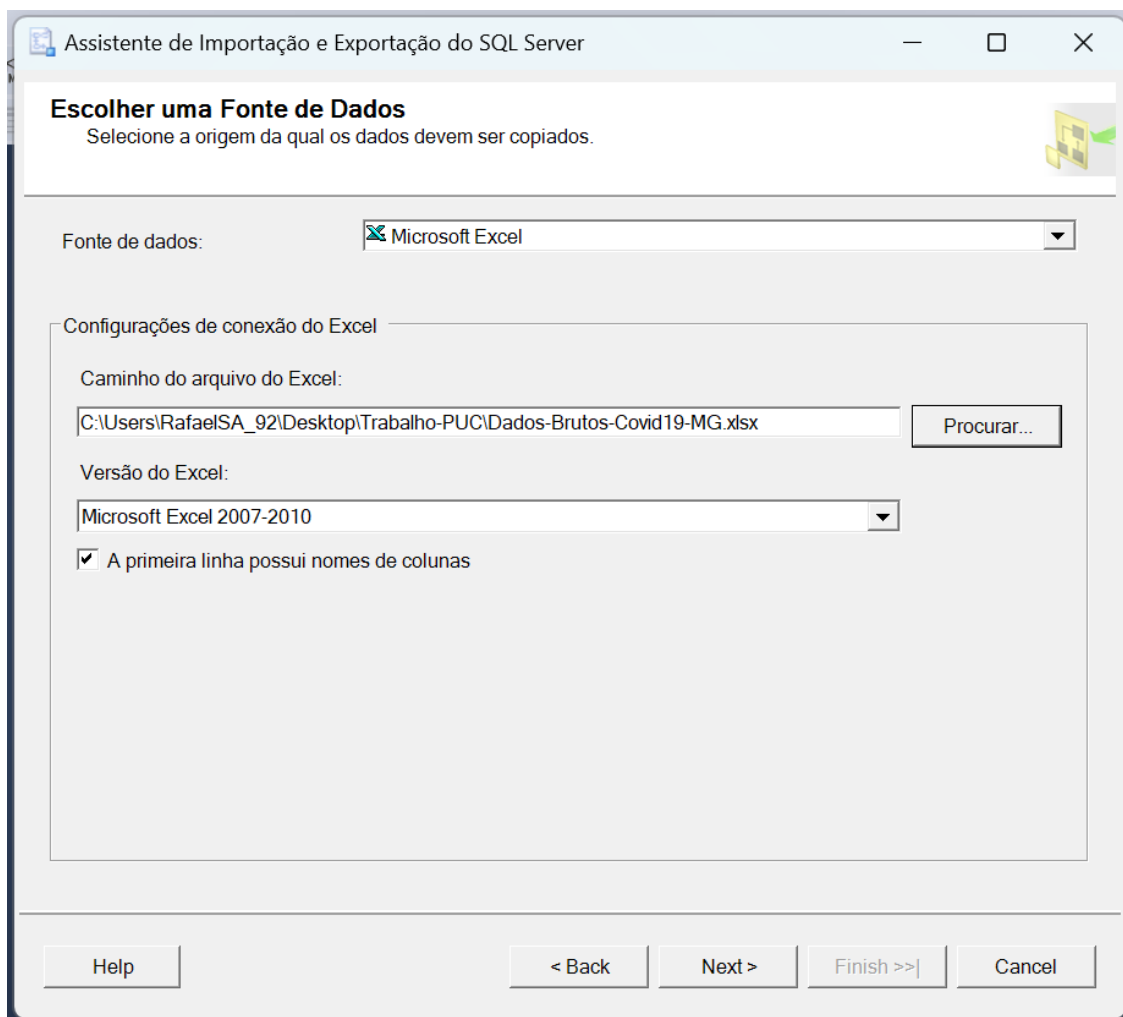


Fonte: Elaborado pelos autores (2023).

Ao abrir o AIESS, na primeira janela que aparecerá (Figura 8), você deve selecionar a fonte de dados desejada, que neste caso é Microsoft Excel, e informar o caminho do arquivo desejado, além da sua versão. O AIESS estabelecerá então uma conexão com o arquivo Excel escolhido, utilizando drivers OLE DB ou ODBC, que são bibliotecas desenvolvidas pela Microsoft que permitem acesso a diversas fontes de dados.

Em seguida, o usuário especifica um destino para onde os dados serão copiados (que, neste caso, é o ‘*SQL Server Native Client*’), o nome do servidor (que aqui será um servidor local, ou seja, a própria máquina do usuário) e a base-de-dados a ser utilizada (que, neste exemplo, é a base foi criada anteriormente).

Figura 8. Janela de seleção da fonte de dados



Fonte: Elaborado pelos autores (2023).

Na janela seguinte (Figura 9), deve-se selecionar a tabela que será copiada e a tabela destino na base-de-dados criada (caso a tabela não exista, ela será criada). Na próxima janela, deve-se selecionar a opção de 'copiar dados de uma ou mais tabelas ou exibições' e, na janela seguinte (Figura 10), pode-se fazer o mapeamento de colunas onde o usuário instrui o AIESS sobre como os dados do Excel devem ser inseridos nas colunas correspondentes da tabela de destino no SQL Server, ou seja, define os tipos de dados em cada coluna, seu tamanho, etc.

Durante o processamento, o AIESS converte tipos de dados, valida informações e insere os dados nas colunas correspondentes. Se a opção de atualização estiver selecionada, ele também gerencia conflitos para evitar duplicações ou sobrescrição de registros existentes.

Figura 9. Janela de definição do destino dos dados

Assistente de Importação e Exportação do SQL Server

Escolher um Destino
Especifique onde os dados devem ser copiados.

Destino: SQL Server Native Client 11.0

Nome do servidor: HAL-9000

Autenticação

☒ Usar Autenticação do Windows

☐ Usar Autenticação do SQL Server

Nome de usuário:

Senha:

Banco de dados: DadosCovid19MG

Atualizar

Novo...

Ajuda < Voltar Avançar > Concluir >> Cancelar

Fonte: Elaborado pelos autores (2023).

Figura 10. Janela de mapeamento de colunas

Assistente de Importação e Exportação do SQL Server

Mapeamentos de Colunas

Origem: Dados_Brutos\$

Destino: [dbo].[Dados_Brutos\$]

☒ Criar tabela de destino Editar SQL...

☐ Excluir linhas na tabela de destino ☐ Ignorar e recriar tabela de destino

☐ Anexar linhas à tabela de destino ☐ Habilitar inserção de identidade

Mapeamentos:

Origem	Destino	Tipo	Permit...	Taman...	Precisão	Esc...
ID	ID	nvarchar	<input checked="" type="checkbox"/>	255		
URS	URS	nvarchar	<input checked="" type="checkbox"/>	255		
MICRO	MICRO	nvarchar	<input checked="" type="checkbox"/>	255		
MACRO	MACRO	nvarchar	<input checked="" type="checkbox"/>	255		
DATA_NOTIFIC...	DATA_NOTIFIC...	nvarchar	<input checked="" type="checkbox"/>	255		
CLASSIFICACA...	CLASSIFICACA...	nvarchar	<input checked="" type="checkbox"/>	255		
SEXO	SEXO	nvarchar	<input checked="" type="checkbox"/>	255		
IDADE	IDADE	nvarchar	<input checked="" type="checkbox"/>	255		
FAIXA ETARIA	FAIXA ETARIA	nvarchar	<input checked="" type="checkbox"/>	255		

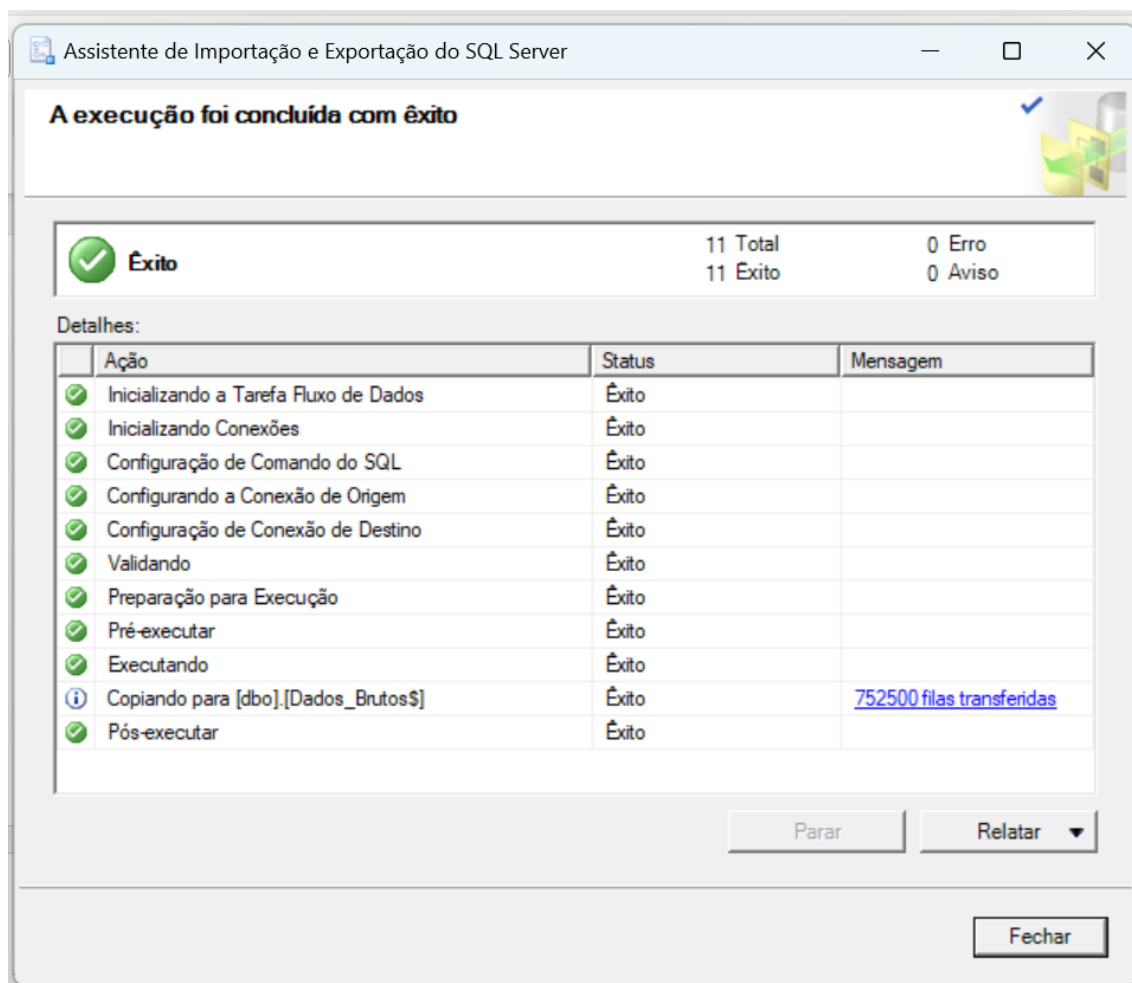
Coluna de origem: ID VarChar (255)

OK Cancelar

Fonte: Elaborado pelos autores (2023).

Ao longo do processo, o AIESS fornece relatórios detalhados sobre o progresso da importação e quaisquer erros encontrados. Essa transparência permite ao usuário acompanhar o status da importação e tomar medidas corretivas, se necessário. Após a conclusão da importação, um resumo é apresentado, indicando quantas linhas foram importadas com sucesso e se ocorreram erros. A Figura 11, a seguir, ilustra essa etapa final do processo de importação pelo AIESS.

Figura 11. Janela de execução do processo de importação



Fonte: Elaborado pelos autores (2023).

Em resumo, o AIESS atua como uma ponte entre o Excel e o SQL Server, simplificando a tarefa de importar dados de arquivos ‘.xlsx’ para um banco de dados, encapsulando os processos, tornando o processo de importação mais acessível e garantindo a integridade dos dados.

5.3 Comparação dos métodos de importação de dados

Comparando-se ambos os métodos de importação de dados para o SGBD, fica claro que o primeiro método exige mais conhecimentos por parte do usuário, que deve saber programar em algum tipo de linguagem de programação e também estar familiarizado

com a estrutura dos dados que serão importados para garantir que o código produzido funcione de forma correta. Por exemplo, apenas depois de algumas tentativas foi possível perceber que alguns da coluna ‘municipio_residencia’ possuíam apóstrofos e isso estava gerando erros na hora de executar o script no SQL Server, pois este usa os apóstrofos para caracterizar os valores do tipo caractere e data. O Assistente de Importação e Exportação, por sua vez, ‘esconde’ do usuário as complexidades envolvidas na inserção dos dados no banco-de-dados, e, portanto, o usuário só precisa saber que botões apertar.

No que se refere à eficiência (rapidez) dos processos, pode-se afirmar que, neste caso, o AIESS também prevalece, uma vez que este programa foi desenvolvido por um time de uma grande empresa, e, portanto, tem desempenho muito maior do que um aplicativo desenvolvido por um estudante. Com o AIESS, foi possível realizar a inserção direta de todos os dados de uma vez, enquanto o programa desenvolvido precisou separar os conjuntos de comandos em oito scripts para que a inserção fosse realizada.

O desenvolvimento de um programa, no entanto, possui algumas vantagens, visto que foi desenvolvido com foco neste conjunto de dados em específico, permitindo, assim, que o processo se molde às necessidades particulares do mesmo. Por exemplo, seria possível e relativamente fácil fazer com que o programa, antes de inserir os dados no banco-de-dados, alterasse valores específicos e adicionasse ou removesse linhas específicas, o que não pode ser feito com o AIESS.

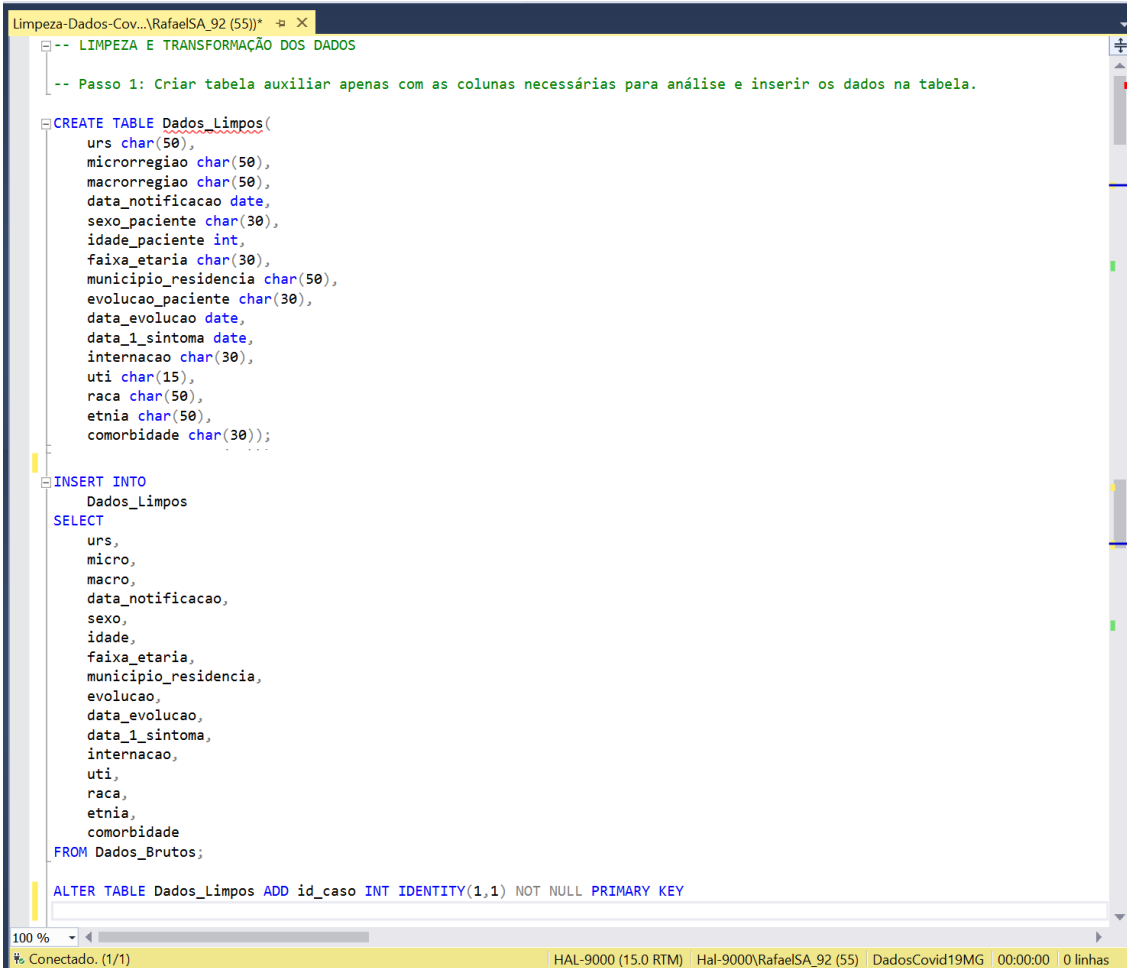
Já no que se refere ao aprendizado, com certeza o desenvolvimento de um programa para inserção de dados tem muito mais valor do que a simples utilização de um assistente, visto que permite ao usuário/desenvolvedor praticar conceitos e entender os processos por trás da inserção.

De qualquer forma, cada um dos métodos tem suas vantagens e desvantagens e situações em que um seria uma melhor opção do que o outros. Para conjuntos grandes e bem estruturados de dados, talvez o AIESS seja uma melhor opção, visto que é mais rápido, mas em conjuntos de dados que precisam ser tratados primeiramente, é provável que a utilização de um programa personalizado traga mais benefícios.

6. Limpeza e Transformação

Na etapa de limpeza e transformação, primeiramente foi criada uma tabela auxiliar chamada ‘Dados_Limpos’, de forma a deixar a tabela principal inalterada, para futuras referências. Após a criação desta tabela usando o comando ‘CREATE TABLE’, todas as colunas da tabela principal ‘Dados_Brutos’, exceto ‘codigo’ (que foi considerada irrelevante para análise) e ‘id_caso’ (que estava inconsistente, com um salto nos valores a partir de determinada linha), foram copiados para a tabela auxiliar, utilizando o comando ‘INSERT’. Em seguida, utilizando-se os comandos ‘ALTER TABLE’ e ‘ADD’, foi inserida uma nova coluna ‘id_caso’, como chave primária da tabela, a fim de identificar cada um dos registros. O código SQL utilizado para realizar essas operações está ilustrado a seguir, na Figura 12:

Figura 12. Código de criação de tabela auxiliar e inserção de dados na mesma



```
-- LIMPEZA E TRANSFORMAÇÃO DOS DADOS
-- Passo 1: Criar tabela auxiliar apenas com as colunas necessárias para análise e inserir os dados na tabela.

CREATE TABLE Dados_Limos(
  urs char(50),
  microrregiao char(50),
  macrorregiao char(50),
  data_notificacao date,
  sexo_paciente char(30),
  idade_paciente int,
  faixa_etaria char(30),
  municipio_residencia char(50),
  evolucao_paciente char(30),
  data_evolucao date,
  data_1_sintoma date,
  internacao char(30),
  uti char(15),
  raca char(50),
  etnia char(50),
  comorbidade char(30));

INSERT INTO
  Dados_Limos
SELECT
  urs,
  micro,
  macro,
  data_notificacao,
  sexo,
  idade,
  faixa_etaria,
  municipio_residencia,
  evolucao,
  data_evolucao,
  data_1_sintoma,
  internacao,
  uti,
  raca,
  etnia,
  comorbidade
FROM Dados_Brutos;

ALTER TABLE Dados_Limos ADD id_caso INT IDENTITY(1,1) NOT NULL PRIMARY KEY
```

Fonte: Elaborado pelos autores (2023).

Após a criação da tabela e inserção dos dados, fez-se uma verificação de cada uma das colunas da tabela, utilizando-se o comando ‘SELECT’ e as cláusulas ‘WHERE’ e ‘FROM’, a fim de identificar a existência de inconsistências. Além disso, ‘DISTINCT’ e ‘ORDER BY’ foram usados em conjunto com ‘SELECT’ para verificar se havia dados com valores discrepantes ou duplicados, com digitação diferente. A partir dessas consultas, percebeu-se a existência de inconsistências nas colunas ‘idade_paciente’, ‘municipio_residencia’, ‘etnia’ e ‘data_1_sintoma’.

Na coluna ‘idade_paciente’, havia 149 linhas com valores maiores que 110, sendo que 123 delas possuíam valores maiores ou iguais a 130, o que não faz sentido, visto que, segundo a Wikipedia, só há 7 casos confirmados de supercentenários no Brasil, e a maior idade registrada é de 116 anos. Desta forma, os campos ‘idade_paciente’ e ‘faixa_etaria’ das linhas cujos valores da idade eram maiores que 110 anos foram trocados por ‘NULL’, mantendo-se o restante dos campos inalterados.

Na coluna ‘municipio_residencia’, foi verificada a existência de valores duplos, mas com digitações diferentes (‘PINGO-D’AGUA’ e ‘PINGO D AGUA’, ‘GUARDA

MOR’ e ‘GUARDA-MOR’, ‘OLHOS D AGUA’ e ‘OLHOS-D’AGUA’, ‘PASSA VINTE’ e ‘PASSA-VINTE’ e ‘SEM PEIXE’ e ‘SEM-PEIXE’). Neste caso, optou-se por manter apenas uma das variantes, substituindo-se todos os registros por uma delas.

O mesmo ocorreu para a coluna ‘etnia’, que possuía três tipos de valores duplicados com digitações diferentes ('KRENAK' e 'KRENAK (BORUN, CRENAQUE)', 'MAXAKALI' e 'MAXAKALI (MAXACALI)', 'XACRIABA', 'XACRIABA (XAKRIABA)' e 'XAKRIABA (XACRIABA)') e uma digitação incorreta ('PATA XO HA-HA-HAE' estava como 'PATA XO-HA-HA-HA'). Como no caso anterior, manteve-se apenas uma das variantes nos casos de duplicação, e no caso de digitação incorreta, os valores foram corrigidos.

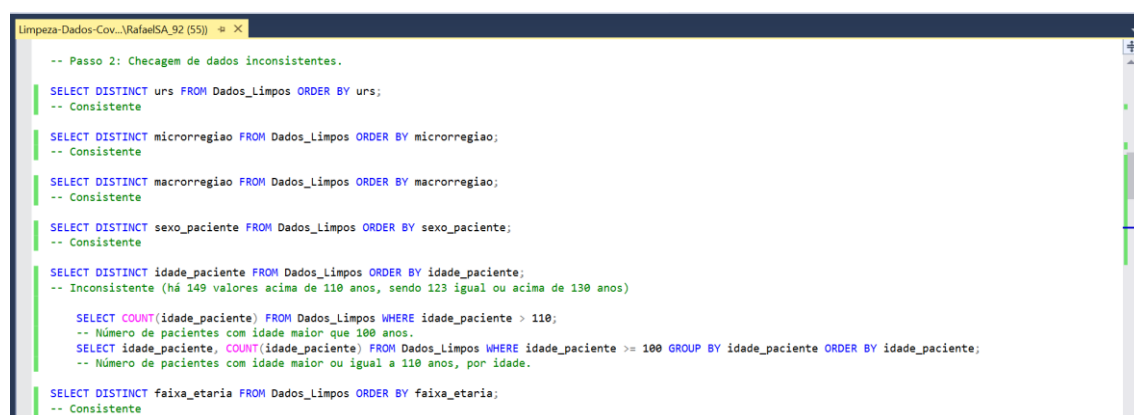
Também na coluna ‘etnia’, havia o valor ‘NAO INDIGENA’ na linha cujo valor da coluna ‘raca’ era ‘INDIGENA’, o que é contraditório. Optou-se por substituir o valor da coluna ‘etnia’ por ‘NULL’, inicialmente.

Ainda na coluna ‘etnia’, achou-se pertinente substituir os valores ‘NULL’ para ‘NÃO INFORMADO’, nas linhas em que o valor da coluna ‘raca’ era igual a ‘INDIGENA’ ou ‘NÃO INFORMADO’, e para ‘NÃO INDIGENA’ nas linhas em que o valor da coluna ‘raca’ fosse igual a ‘BRANCO’, ‘PARDO’, ‘AMARELO’ ou ‘PRETO’ (ou seja, diferente de ‘INDIGENA’ ou ‘NÃO DEFINIDO’).

Na coluna ‘data_1_sintoma’, havia uma linha com valor ‘2121-05-10’, que foi substituído por ‘2021-05-10’, e uma linha com valor ‘1970-01-01’, o qual foi substituído por ‘NULL’.

A seguir, na Figura 13 e na Figura 14, respectivamente, são mostradas as consultas realizadas para verificar e corrigir as inconsistências nos dados:

Figura 13. Consultas para verificação de inconsistências nos dados



```
-- Passo 2: Checagem de dados inconsistentes.

SELECT DISTINCT urs FROM Dados_Limos ORDER BY urs;
-- Consistente

SELECT DISTINCT microrregiao FROM Dados_Limos ORDER BY microrregiao;
-- Consistente

SELECT DISTINCT macrorregiao FROM Dados_Limos ORDER BY macrorregiao;
-- Consistente

SELECT DISTINCT sexo_paciente FROM Dados_Limos ORDER BY sexo_paciente;
-- Consistente

SELECT DISTINCT idade_paciente FROM Dados_Limos ORDER BY idade_paciente;
-- Inconsistente (há 149 valores acima de 110 anos, sendo 123 igual ou acima de 130 anos)

SELECT COUNT(idade_paciente) FROM Dados_Limos WHERE idade_paciente > 110;
-- Número de pacientes com idade maior que 100 anos.
SELECT idade_paciente, COUNT(idade_paciente) FROM Dados_Limos WHERE idade_paciente >= 100 GROUP BY idade_paciente ORDER BY idade_paciente;
-- Número de pacientes com idade maior ou igual a 110 anos, por idade.

SELECT DISTINCT faixa_etaria FROM Dados_Limos ORDER BY faixa_etaria;
-- Consistente
```

```

SELECT DISTINCT municipio_residencia FROM Dados_Limos ORDER BY municipio_residencia;
-- Inconsistente
--(Valores inconsistentes: PINGO-D'AGUA E PINGO D AGUA, GUARDA MOR E GUARDA-MOR, OLHOS D AGUA E OLHOS-D'AGUA, PASSA VINTE E PASSA-VINTE, SEM PEIXE e SEM-PEIXE).

SELECT DISTINCT internacao FROM Dados_Limos ORDER BY internacao;
-- Consistente

SELECT DISTINCT evolucao_paciente FROM Dados_Limos ORDER BY evolucao_paciente;
-- Consistente

SELECT DISTINCT uti FROM Dados_Limos ORDER BY uti;
-- Consistente

SELECT DISTINCT raca FROM Dados_Limos ORDER BY raca;
-- Consistente

SELECT DISTINCT etnia FROM Dados_Limos ORDER BY etnia;
-- Inconsistente
-- Valores duplicados ('KRENAK' E 'KRENAK(BORUN, CRENAQUE)', 'MAXAKALI' E 'MAXAKALI(MAXACALI)', 'XACRIABA' E 'XACRIABA(XAKRIABA)' E 'XAKRIABA(XAKRIABA)').
-- PATAXO HA-HA-HAE está como PATAXO-HA-HA-HA.

SELECT DISTINCT etnia, raca FROM Dados_Limos ORDER BY etnia;
-- Inconsistente (Há uma linha com raça 'INDIGENA' e etnia 'NAO INDIGENA').

SELECT DISTINCT comorbidade FROM Dados_Limos ORDER BY comorbidade;
-- Consistente

SELECT DISTINCT data_notificacao FROM Dados_Limos ORDER BY data_notificacao;
-- Consistente

SELECT DISTINCT data_evolucao FROM Dados_Limos ORDER BY data_evolucao;
-- Consistente

SELECT DISTINCT data_1_sintoma FROM Dados_Limos ORDER BY data_1_sintoma
-- Inconsistente (Um valor como 2121-05-10 e um valor como 01-01-1970).

SELECT * FROM Dados_Limos WHERE data_1_sintoma = '2121-05-10';
SELECT * FROM Dados_Limos WHERE data_1_sintoma = '1970-01-01';

```

Fonte: Elaborado pelos autores (2023).

Figura 14. Consultas para correção de inconsistências nos dados para correção de dados inconsistentes

```

-- Passo 3: Correção de inconsistências dos dados

-- Remoção campos de idade_paciente e faixa_etaria das linhas de casos com idade do paciente maior que 110 anos.
UPDATE Dados_Limos SET idade_paciente = NULL, faixa_etaria = NULL WHERE idade_paciente > 110;

-- Substituição dos valores duplicados
UPDATE Dados_Limos SET municipio_residencia = 'PINGO D AGUA' WHERE municipio_residencia = 'PINGO-D'AGUA';
UPDATE Dados_Limos SET municipio_residencia = 'GUARDA-MOR' WHERE municipio_residencia = 'GUARDA MOR';
UPDATE Dados_Limos SET municipio_residencia = 'OLHOS D AGUA' WHERE municipio_residencia = 'OLHOS-D'AGUA';
UPDATE Dados_Limos SET municipio_residencia = 'PASSA-VINTE' WHERE municipio_residencia = 'PASSA VINTE';
UPDATE Dados_Limos SET municipio_residencia = 'SEM-PEIXE' WHERE municipio_residencia = 'SEM PEIXE';
UPDATE Dados_Limos SET etnia = 'KRENAK (BORUN, CRENAQUE)' WHERE etnia = 'KRENAK';
UPDATE Dados_Limos SET etnia = 'MAXAKALI (MAXACALI)' WHERE etnia = 'MAXAKALI';
UPDATE Dados_Limos SET etnia = 'XAKRIABA (XAKRIABA)' WHERE etnia = 'XACRIABA' OR etnia = 'XACRIABA (XAKRIABA)';
UPDATE Dados_Limos SET etnia = 'PATAXO HA-HA-HAE' WHERE etnia = 'PATAXO HA-HA-HA' OR etnia = 'PATAXO';

-- Correção das datas inconsistentes
UPDATE Dados_Limos SET data_1_sintoma = '2021-05-10' WHERE data_1_sintoma = '2121-05-10';
UPDATE Dados_Limos SET data_1_sintoma = NULL WHERE data_1_sintoma = '1970-01-01';

-- Ajuste dos valores de raca e etnia
UPDATE Dados_Limos SET etnia = NULL WHERE raca = 'INDIGENA' AND etnia = 'NAO INDIGENA';
UPDATE Dados_Limos SET etnia = 'NAO INDIGENA' WHERE raca != 'INDIGENA' AND raca != 'NAO INFORMADO';
UPDATE Dados_Limos SET etnia = 'NAO INFORMADO' WHERE raca = 'INDIGENA' AND etnia IS NULL;
UPDATE Dados_Limos SET etnia = 'NAO INFORMADO' WHERE raca = 'NAO INFORMADO';

```


Fonte: Elaborado pelos autores (2023).

Não foi necessária a realização de alteração dos nomes das colunas ou adequação dos tipos dos dados, pois já haviam sido realizadas durante a importação dos dados, quando necessário.

7. Visualização dos Dados

Uma vez realizado o processo de ETL, é possível, ainda utilizando-se a Linguagem SQL e o SQL Server, com auxílio do SSMS, fazer uma análise exploratória dos dados. Abaixo, na Figura 15, são ilustrados exemplos de perguntas, as expressões SQL utilizadas para respondê-las e as respostas das perguntas realizadas:

Figura 15. Exemplos de análise exploratória de dados com SQL, SQL Server e SSMS



```
-- Análise Exploratória dos Dados

-- P: Qual o número de casos no ano de 2020?
SELECT COUNT (*) FROM Dados_Limpas WHERE YEAR (data_notificacao) = 2020;
-- R: 322.644.

-- P: Qual número de casos no ano de 2021?
SELECT COUNT (*) FROM Dados_Limpas WHERE YEAR (data_notificacao) = 2021;
-- R: 429.438.

-- P: Qual a unidade regional de atendimento com maior número de casos?
SELECT urs, COUNT (*) AS numero_de_casos FROM Dados_Limpas WHERE urs IS NOT NULL GROUP BY urs ORDER BY numero_de_casos DESC;
-- R: Belo Horizonte, com 185573 casos.

-- P: Qual a microrregião com maior número de casos?
SELECT microrregiao, COUNT (*) AS numero_de_casos FROM Dados_Limpas WHERE microrregiao IS NOT NULL GROUP BY microrregiao ORDER BY numero_de_casos DESC;
-- R: Belo Horizonte/Nov Lima/Caete, com 42593 casos.

-- P: Qual a macrorregião com maior número de casos?
SELECT macrorregiao, COUNT (*) AS numero_de_casos FROM Dados_Limpas WHERE macrorregiao IS NOT NULL GROUP BY macrorregiao ORDER BY macrorregiao;
-- R: Centro, com 160502 casos.

-- P: Qual o número de casos por evolução do paciente?
SELECT evolucao_paciente, COUNT (evolucao_paciente) AS numero_de_casos FROM Dados_Limpas GROUP BY evolucao_paciente ORDER BY evolucao_paciente;
-- R: Em acompanhamento (76301), Óbito (45036), Recuperado (631163).

-- P: Qual a faixa etária com maior número de casos?
SELECT faixa_etaria, COUNT (faixa_etaria) AS numero_de_casos
FROM Dados_Limpas
WHERE faixa_etaria IS NOT NULL GROUP BY faixa_etaria ORDER BY numero_de_casos DESC;
-- R: 30 a 39 anos, com 158655 casos.

-- P: Qual a faixa etária com maior número de óbitos?
SELECT faixa_etaria, evolucao_paciente, COUNT (evolucao_paciente) AS numero_de_casos
FROM Dados_Limpas
WHERE faixa_etaria IS NOT NULL AND evolucao_paciente = 'OBITO' GROUP BY faixa_etaria, evolucao_paciente ORDER BY numero_de_casos DESC;
-- R: 70 a 79 anos, com 10967 casos.

-- P: Qual o número de óbitos por comorbidade?
SELECT comorbidade, evolucao_paciente, COUNT (evolucao_paciente) AS numero_de_casos
FROM Dados_Limpas
WHERE comorbidade IS NOT NULL AND comorbidade != 'NAO INFORMADO' AND evolucao_paciente = 'OBITO'
GROUP BY comorbidade, evolucao_paciente ORDER BY comorbidade, evolucao_paciente;
-- R: Com comorbidade (38510), Sem comorbidade (14526).

-- P: Qual o número de internações em UTI por comorbidade?
SELECT comorbidade, uti, COUNT (uti) AS numero_de_casos
FROM Dados_Limpas
WHERE comorbidade IS NOT NULL AND comorbidade != 'NAO INFORMADO' AND uti = 'SIM'
GROUP BY comorbidade, UTI ORDER BY comorbidade;
-- R: Com comorbidade (27034), Sem comorbidade (15546).

-- P: Qual o número de casos por raça?
SELECT raca, COUNT (raca) AS numero_de_casos
FROM Dados_Limpas
GROUP BY raca ORDER BY numero_de_casos;
-- R: Parda (271838), Branca (270904), Preta (38057), Amarela (26875), Indígena (540).

-- Qual o número de óbitos por raça?
SELECT raca, evolucao_paciente, COUNT (evolucao_paciente) AS numero_de_casos
FROM Dados_Limpas
WHERE evolucao_paciente = 'OBITO' GROUP BY raca, evolucao_paciente ORDER BY numero_de_casos DESC;
-- R: Parda (18390), Branca (19363), Preta (3702), Amarela (296), Indígena (18).

-- Quantos e Quais são os municípios com mais de 20000 residentes atendidos, em ordem decrescente de numero de casos?
SELECT municipio_residencia, COUNT (municipio_residencia) AS numero_de_casos
FROM Dados_Limpas
GROUP BY municipio_residencia HAVING COUNT (municipio_residencia) >= 20000 ORDER BY COUNT (municipio_residencia) DESC;
-- R: 5 Municípios. Ipatinga (27522), Belo Horizonte (27155), Contagem (26393), Uberlândia (22482), Governador Valadares (21311).

-- Quais as etnias (Povos indígenas) com maior número de casos?
SELECT etnia, COUNT (etnia) AS numero_de_casos FROM Dados_Limpas WHERE etnia != 'NAO INDIGENA' GROUP BY etnia ORDER BY numero_de_casos DESC;
-- R: Xakriaba (172), Maxacali (89), Pataxo Ha-Ha-Ha (47), Krenak (36) e Kaxixó (10).
```

Fonte: Elaborado pelos autores (2023).

Optou-se, entretanto, no presente trabalho, por fazer a análise e visualização dos dados pelo Power BI. Para isso, após a realização do processo de ETL, os dados foram importados para o Power BI, via conexão com a base de dados criada no SQL Server, para realização da análise dos dados e geração das visualizações.

O primeiro passo foi criar, utilizando-se a função ‘CALENDARAUTO’ da linguagem DAX, já mencionada, uma tabela dimensão ‘calendario’, contendo todas as datas de notificação da tabela original, que foi então conectada à tabela original por meio dos campos ‘date’, da tabela ‘calendario’, e ‘data_notificacao’, da tabela original. Ainda utilizando a linguagem DAX, foram criadas uma série de medidas, a fim de permitir a criação das visualizações pretendidas.

A seguir, na Tabela 2, constam as medidas que foram criadas e as expressões DAX utilizadas em sua criação:

Tabela 1. Estrutura da base de dados utilizada na íntegra.

Medida Criada	Expressão DAX utilizada
% Casos Indigenas	% Casos Indigenas = DIVIDE([Qtde Casos Raca Indigena], COUNTROWS(ALL(Dados_Limpos)))
Qtde Casos Raca Indigena	Qtde Casos Raca Indigena = CALCULATE(IF(ISBLANK(DISTINCTCOUNT(Dados_Limpos[id_caso])),0, DISTINCTCOUNT(Dados_Limpos[id_caso])), Dados_Limpos[raca] = "INDIGENA")
Qtde de Casos	Qtde de Casos = DISTINCTCOUNT(Dados_Limpos[id_caso])
Qtde de Internacoes	Qtde de Internações = CALCULATE(DISTINCTCOUNT(Dados_Limpos[id_caso]), Dados_Limpos[internacao]= "SIM")
Qtde de UTI	Qtde de UTI = CALCULATE(DISTINCTCOUNT(Dados_Limpos[id_caso]), Dados_Limpos[uti]= "SIM")
Qtde de Obitos	Qtde Obitos = CALCULATE(DISTINCTCOUNT(Dados_Limpos[id_caso]), Dados_Limpos[evolucao_paciente]= "OBITO")
Qtde de Recuperacoes	Qtde Recuperacoes = CALCULATE(DISTINCTCOUNT(Dados_Limpos[id_caso]), Dados_Limpos[evolucao_paciente]= "RECUPERADO")

Fonte: Elaborado pelos autores (2023).

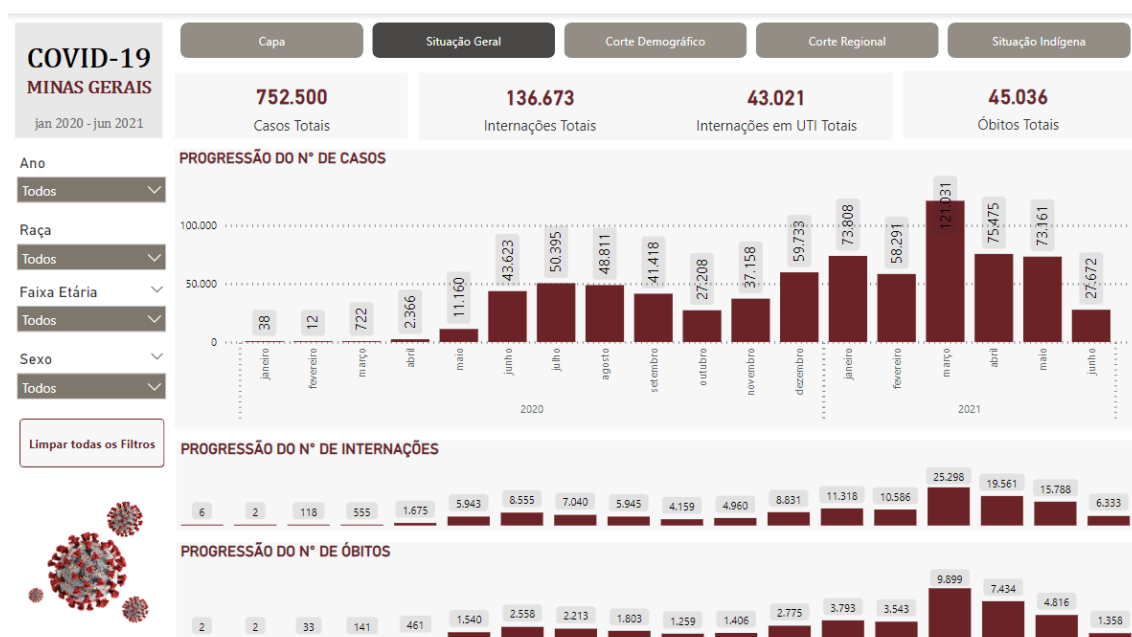
Uma vez realizados esses passos iniciais, procedeu-se à criação dos *Dashboards*, os quais são painéis que contém informações, métricas e indicadores pertinentes à análise de dados de uma organização. A seguir, são apresentados cada um dos painéis desenvolvidos, explicado o seu processo de criação e em seguida é feita a interpretação dos resultados obtidos com a análise dos mesmos.

7.1 Painel de Situação Geral

Este painel contém as métricas e indicadores relativos a todos os casos registrados de Covid-19 no Estado de Minas Gerais entre janeiro de 2020 e junho de 2021. Nele é possível ver o número total de casos do período, além do total de internações, internações em UTI e o número total de óbitos.

Além destas informações, o painel contém gráficos com a progressão do número de casos, internações e óbitos mês-a-mês. É possível, ainda, fazer uma filtragem dos dados por ano, além de raça, faixa etária e sexo dos pacientes. Abaixo, na Figura 16, tem-se a ilustração do Painel de Situação Geral:

Figura 16. Painel de Situação Geral



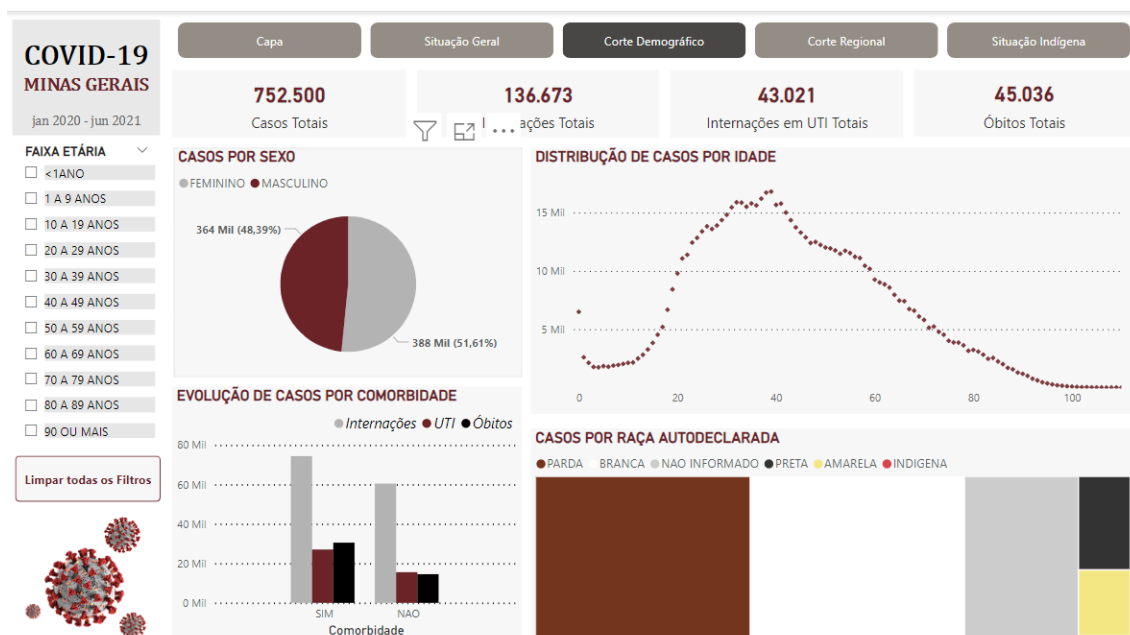
Fonte: Elaborado pelos autores (2023).

A análise das informações geradas por este painel nos permite afirmar que o total internações foi igual a 136.673 e o total de óbitos foi igual a 45.036 no período de dezoito meses analisado, sendo que os picos de internações e óbitos ocorreram nos meses julho de 2020, na primeira ‘onda’ e em março de 2021, na segunda ‘onda’.

7.2 Painel de Corte Demográfico

Este painel contém informações sobre os dados da Covid-19 em Minas Gerais sob uma perspectiva demográfica. Nele, é possível observar a distribuição dos casos por sexo, idade, raça e comorbidade. Também é possível fazer uma filtragem por faixa etária dos pacientes. A seguir, na Figura 17, tem-se a ilustração deste painel:

Figura 17. Painel de Corte Demográfico



Fonte: Elaborado pelos autores (2023).

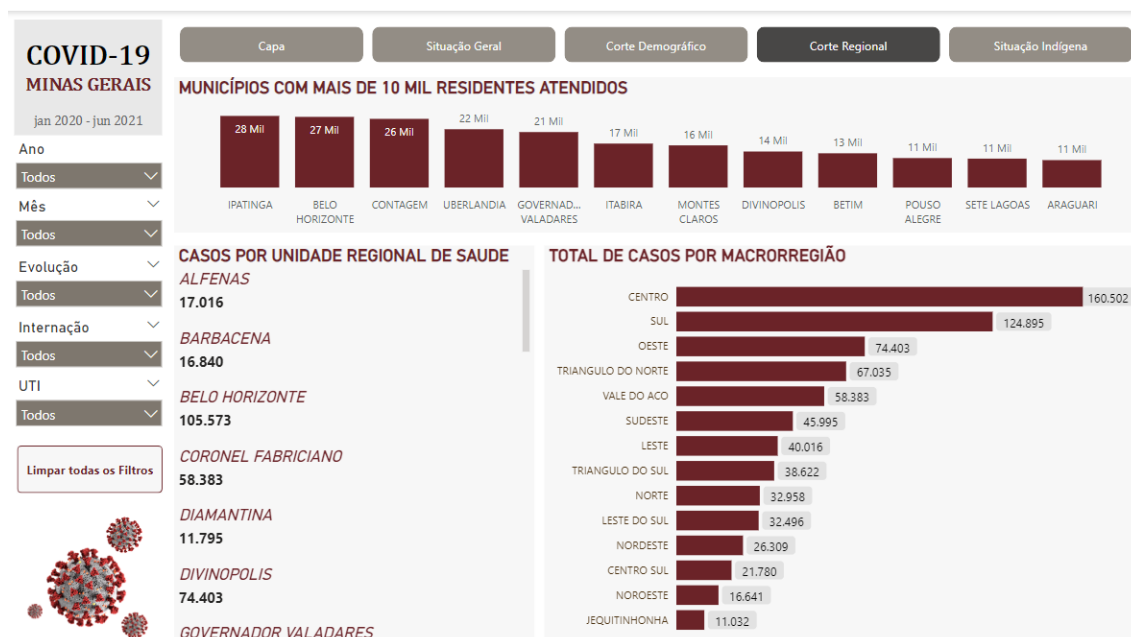
É possível perceber, a partir da observação dos gráficos contidos neste painel, que ambos os sexos foram afetados de forma similar, apresentando aproximadamente 50% dos casos registrados cada um. Percebe-se, também, que o número de casos foi maior entre os adultos entre 25 e 45 anos, possivelmente porque este é o grupo com maior densidade populacional.

No que se refere à comorbidade, percebe-se que os pacientes que afirmaram possuir alguma comorbidade apresentaram quase o dobro de internações em UTI dos que não afirmaram possuir (27 mil contra 15 mil) e mais que o dobro de óbitos (30 mil contra 14 mil). Em se tratando da raça, percebe-se que os casos entre os autodeclarados pardos e brancos foi aproximadamente o mesmo, sendo bem maiores que os autodeclarados pretos ou amarelos, enquanto a quantidade de casos entre os autodeclarados indígenas foi tão pequena que não aparece no gráfico. Isso se justifica, possivelmente, novamente, pela distribuição demográfica da população brasileira.

7.3 Painel de Corte Geográfico

Neste painel, é feita uma análise dos dados sob uma perspectiva geográfica, mostrando-se, ao topo, os municípios com mais de dez mil habitantes atendidos, ao centro esquerdo, uma lista com o número de casos por Unidade Regional de Atendimento e, finalmente, ao centro direito, uma tabela de barras com o total de casos por macrorregião. É possível fazer filtragem por ano, mês, evolução do paciente, além de internações e internações em UTI. A seguir, na Figura 18, tem-se a ilustração deste painel:

Figura 18. Painel de Situação Geral



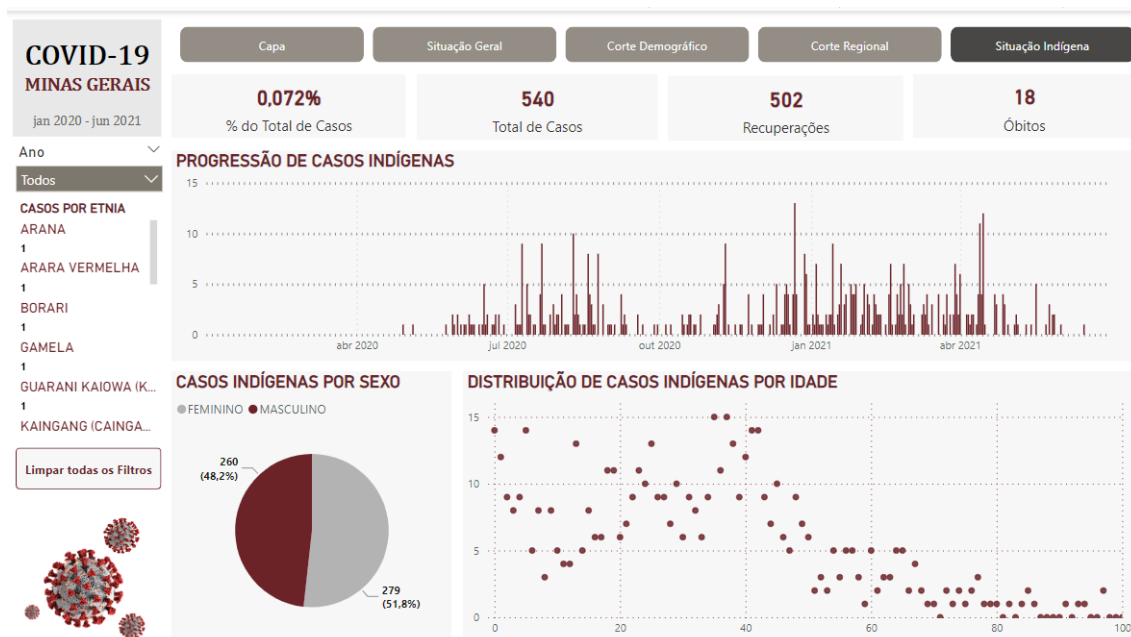
Fonte: Elaborado pelos autores (2023).

Neste painel, vemos que os municípios com maior número de residentes atendidos foram Ipatinga, Belo Horizonte e Contagem, e as macrorregiões com maior número de casos foram a macrorregião Centro e Sul.

7.4 Painel de Situação Indígena

Neste painel, que é o último, temos informações referentes aos casos de pacientes autodeclarados indígenas. Nele, é possível observar a progressão temporal dos casos entre a população indígena atendida em Minas Gerais, além da distribuição de casos por sexo e idade. É possível, também, fazer uma filtragem por etnia. A seguir, na Figura 19, tem-se a ilustração deste painel:

Figura 19. Painel de Situação Indígena



Fonte: Elaborado pelos autores (2023).

Pela análise das informações contidas neste painel, podemos que não houve, assim, como na análise dos casos com um todo, grande diferença do número de casos entre os sexos, mas percebe-se que houve um maior número de casos entre os indígenas com menos de 60 anos. As etnias com maior número de casos foi a etnia Xakriaba, com 172 casos e a etnia Maxacali, com 89 casos.

8 Conclusão

O tratamento e a análise dos dados fornecidos pelo Governo Federal, em parceria com a Secretaria de Saúde do Estado de Minas Gerais, sobre a Covid-19, no Estado de Minas Gerais, os quais elencaram uma variedade de características da população, tais como sexo, idade, faixa etária, município de residência, raça, etnia e possíveis existências de comorbidades, além de outros aspectos, como local de atendimento do paciente e agravamento do caso que pode ter ocasionado em internação comum ou UTI, permitiram de forma satisfatória, entender os impactos sociodemográficos da doença no referido estado.

Foi possível constatar, por exemplo, que as macrorregiões com maior número de casos registrados foram a macrorregião Centro com 160.502 casos, e a macrorregião Sul, com 124.895 casos, que somadas, representam aproximadamente 38% do total de número de casos. Além disso, foi possível inferir que o sexo, a raça e a idade não foram fatores determinantes na contaminação de pacientes, uma vez que a distribuição das ocorrências em ambos os sexos é equivalente (51,6% masculino e 48,4% feminino), e a distribuição de casos por raça e idade está coerente coma distribuição de pessoas por raça e idade na população. No que se refere a comorbidade, no entanto, os dados parecem apontá-los como fatores relevantes para o agravamento da doença, visto que o número de casos de

internações e óbitos observada entre os pacientes com comorbidade são quase o dobro do número de casos dos pacientes sem comorbidade.

Os *dashboards* produzidos, portanto, podem ser considerados de grande valor para qualquer pessoa que tenha interesse em entender o desenvolvimento da pandemia de Covid-19 no Estado de Minas Gerais e seus impactos na população, mostrando, também, de que forma as populações indígenas foram atingidas.

No que tange a trabalhos futuros, em linhas gerais, verificou-se a metodologia utilizada pelo trabalho pode ser aplicada à construção de bancos de dados e *dashboards* que não se restringem somente ao tema estudado, posto que a infraestrutura adotada atende os propósitos de construção de bancos de dados locais e que, inclusive, podem ser utilizados como fontes para ferramentas de ‘*dataviz*’ como foi o caso do uso do Power BI.

Referências

COVID-19: MG deve receber mais de 400 mil doses de vacina bivalente. Estado de Minas. 03 mai. 2023. Disponível em: https://www.em.com.br/app/noticia/gerais/2023/05/03/interna_gerais,1489383/Covid-19-mg-deve-receber-mais-de-400-mil-doses-de-vacina-bivalente.shtml. Acesso em: 12 set. 2023.

MARIZ, Clara. COVID: Minas Gerais teve 209 novos casos por dia em agosto. Estado de Minas. 31 ago. 2023. Disponível em: https://www.em.com.br/app/noticia/gerais/2023/08/31/interna_gerais,1555113/covid-minas-gerais-teve-209-novos-casos-por-dia-em-agosto.shtml. Acesso em: 12 set. 2023.

MORAIS, Lucas. Vacina contra a Covid: os números da desigualdade em Minas. O Tempo. 18 jun. 2021. Disponível em: <https://www.otempo.com.br/cidades/vacina-contr-a-covid-os-numeros-da-desigualdade-em-minas-1.2500625>. Acesso em: 12 set 2023.

WIKIPEDIA. Lista de supercentenários brasileiros. Disponível em: https://pt.wikipedia.org/wiki/Lista_de_supercenten%C3%A1rios_brasileiros. Acesso em: 13 nov. 2023.