UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Criação de um data warehouse para dados públicos de atendimentos ambulatoriais do SUS

Danilo Gouvea Silva

Monografia - MBA em Inteligência Artificial e Big Data

ICMC USP SÃO CARLOS

Danilo Gouvea Silva

Criação de um data warehouse para dados públicos de atendimentos ambulatoriais do SUS

Monografia apresentada ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Ricardo Araújo Rios

Versão original

São Carlos

2024

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

Silva, Danilo Gouvea
S586c Criação de um data warehouse para dados públicos de atendimentos ambulatoriais do SUS / Danilo Gouvea Silva; orientador Ricardo Araújo Rios. -- São Carlos, 2024.
114 p.

Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2024.

1. Data Warehouse. 2. ETL. 3. Big Data. 4. Spark. 5. Dados Públicos de Saúde. I. Rios, Ricardo Araújo, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Danilo Gouvea Silva

Creation of a data warehouse using outpatient care public data from the brazilian public healthcare system (SUS)

Monograph presented to the Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, as part of the requirements for obtaining the title of Specialist in Artificial Intelligence and Big Data.

Concentration area: Artificial Intelligence

Advisor: Prof. Dr. Ricardo Araújo Rios

Original version

São Carlos

2024

À minha mãe, Vandineuza (in memoriam), que me ensinou desde muito cedo o valor da educação e que trabalhou incansavelmente para que eu tivesse as oportunidades que ela não teve.

AGRADECIMENTOS

Inicio agradecendo as professoras e professores do MBA em Inteligência Artificial e Big Data do ICMC-USP. Fizeram valer cada hora investida nessa especialização. Há anos não me sentia aprendendo tanto.

Agradeço também meu orientador, Professor Ricardo A. Rios, pelo seu suporte na elaboração do meu trabalho de conclusão e na produção dessa monografia. Suas observações sempre muito valiosas.

Aos colegas da turma 2 do MBA, minha turma original, também deixo um enorme obrigado. As constantes e valiosas trocas no Discord da turma tornaram todo o processo muito mais agradável. Desejo sucesso a todos.

Não posso me esquecer dos amigos que ofereceram encorajamento e, quando necessário, puxões de orelha para que esse trabalho fosse concluído com sucesso. Eles sabem quem são.

Finalmente, um agradecimento especial à minha namorada, Rebeca, pela compreensão, apoio e carinho durante a etapa final desse trabalho.

"Sem dados, você é apenas mais uma pessoa com uma opinião."
W. Edwards Deming

RESUMO

Silva, D. G. **Criação de um data warehouse para dados públicos de atendimentos ambulatoriais do SUS**. 2024. 114p. Monografia (MBA em Inteligência Artificial e Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

Os dados da produção ambulatorial do Sistema Único de Saúde (SUS) são armazenados no Sistema de Informações Ambulatoriais (SIA) e são disponibilizados para o público geral para *download* pelo Departamento de Informática do SUS (DATASUS). Esses dados, mesmo anonimizados e sem nenhum campo identificador único de paciente, oferecem grande oportunidade de geração de conhecimento sobre as características dos pacientes, doenças, procedimentos realizados e o custo desses procedimentos para o SUS. O enorme volume de dados armazenados no SIA pode ser classificado como *Big Data*, mas os formatos de arquivo no qual esse dados são disponibilizados e a ferramenta de análise oferecida pelo DATASUS não possibilitam uma análise eficiente no contexto de grandes volumes de dados. O objetivo desse trabalho é criar um *data warehouse* para os dados do SIA, utilizando o motor de processamento distribuído Apache Spark e o formato de arquivo colunar Apache Parquet. Dados brutos do SIA e seus dados auxiliares passaram por processo de ETL e os dados foram organizados, de acordo com a teoria estabelecida sobre *data warehousing*, em um modelo dimensional com uma tabela de fatos e doze tabelas de dimensão. Com os dados transformados e armazenados no data warehouse, consultas analíticas de alta complexidade puderam ser executadas de maneira eficiente e retornaram resultados em tempo hábil.

Palavras-chave: Data Warehouse. ETL. Big Data. Spark. Dados Públicos de Saúde. DATASUS.

ABSTRACT

Silva, D. G. **Creation of a data warehouse using outpatient care public data from the brazilian public healthcare system (SUS)**. 2024. 114p. Monograph (MBA in Artificial Intelligence and Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

Data from outpatient production within the Unified Health System (Sistema Único de Saúde - SUS) is stored in the Ambulatory Information System (Sistema de Informações Ambulatoriais - SIA) and is made available for the general public for download by the Department of Informatics of SUS (DATASUS). Even though this data is anonymized and lacks any unique patient identifier, it presents a significant opportunity to generate knowledge about patient characteristics, diseases, procedures performed, and the costs of these procedures to SUS. The huge amount of data stored in the SIA can be classified as Big Data, but the file format in which this data is provided and the analysis tool offered by DATASUS do not allow for efficient analysis in the context of large data volumes. The aim of this work is to create a data warehouse for the SIA data, utilizing the distributed processing engine Apache Spark and the columnar file format Apache Parquet. Raw data from the SIA and its auxiliary data underwent an ETL process, then the data was organized, following established theories on data warehousing, into a dimensional model with one fact table and twelve dimension tables. With the data transformed and stored in the data warehouse, high-complexity analytical queries could be executed efficiently, yielding timely results.

Keywords: Data warehouse. ETL. Big Data. Spark. Public Healthcare Open Data. Pyspark. DATASUS.

LISTA DE FIGURAS

Figura 1 – Arquitetura de um data warehouse tradicional. (BRITO, 2017) 28
Figura 2 – *Star Schema* ou Modelo Dimensional. (KIMBALL; ROSS, 2013) 30
Figura 3 – Exemplo da junção da tabela de fatos com as tabelas de dimensão. (KIMBALL; ROSS, 2013) 31
Figura 4 – Fluxo de ingestão de dados do CIDACS. (BARRETO *et al.*, 2019) . . . 34
Figura 5 – Seleção dos atributos de interesse da tabela SIA. Elaboração do autor. 39
Figura 6 – Diagrama do modelo dimensional do DW. Apenas chaves primárias sendo mostradas nas tabelas de dimensão. Elaboração do autor. 40
Figura 7 – Exemplo de consulta analítica utilizando o modelo dimensional. Elaboração do autor. 41
Figura 8 – Tempo de execução da consulta de exemplo de acordo com a interface de usuário do Apache Spark. Elaboração do Autor. 41

LISTA DE TABELAS

Tabela 1 – Bases de dados do Desafio Datathon utilizadas nesse trabalho. Elaboração do autor. 36
Tabela 2 – Tabelas de fatos e dimensões. Elaboração do autor. 38

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface* (Interface de Programação da Aplicação)
APAC	Autorização de Procedimentos Ambulatoriais de Alta Complexidade
BI	*Business Intelligence* (Inteligência do Negócio)
CID	Código Internacional de Doenças
CIDACS	Centro de Integração de Dados e Conhecimentos para Saúde
CRM	*Customer Relationship Management* (Gerenciamento de Relacionamento com Clientes)
CNES	Cadastro Nacional de Estabelecimentos de Saúde
DAG	*Directed Acyclic Graph* (Grafo Acíclico Dirigido)
DATASUS	Departamento de Informática do Sistema Único de Saúde
DW	Data Warehouse
ERP	*Enterprise Resource Planning* (Planejamento de Recursos da Empresa)
ETL	*Extract, Transform and Load* (Extração, Transformação e Carregamento)
IBGE	Instituto Brasileiro de Geografia e Estatística
MPP	*Massive Parallel Processing* (Processamento Massivo Paralelo)
NIS	Número de Identificação Social
PBF	Programa Bolsa Família
RDD	*Resilient Distributed Dataset* (Conjunto de Dados Distribuído e Resiliente)
SIA	Sistema de Informações Ambulatoriais do SUS
SIGTAP	Sistema de Gerenciamento da Tabela de Procedimentos
SIH	Sistema de Informações Hospitalares do SUS
SIM	Sistema de Informações sobre Mortalidade

SINAN	Sistema de Informação de Agravos de Notificação
SISNAC	Sistema de Informação sobre Nascidos Vivos
SQL	*Structured Query Language* (Linguagem de Consulta Estruturada)
SUS	Sistema Único de Saúde

SUMÁRIO

1 INTRODUÇÃO 25

2 FUNDAMENTAÇÃO TEÓRICA 27

2.1 O Data Warehouse Tradicional 27

2.1.1 Arquitetura do Data Warehouse 28

2.1.1.1 Fontes de Dados 28

2.1.1.2 *ETL - Extract, Load and Transform* 28

2.1.1.3 Data Warehouse 29

2.1.1.4 Aplicações de *Business Intelligence* 29

2.1.2 Modelagem Dimensional 29

2.1.2.1 Tabelas de Fatos para Medições 30

2.1.2.2 Tabelas de Dimensão para Contexto Descritivo 30

2.2 Tecnologias de Big Data 31

2.2.1 MPP ou Processamento Paralelo Massivo 31

2.2.2 Armazenamento colunar e Apache Parquet 32

2.2.3 Apache Spark 32

3 TRABALHOS RELACIONADOS 33

4 PROPOSTA - FONTES DE DADOS E INFRAESTRUTURA 35

4.1 Sistema de Informações Ambulatoriais (SIA) 35

4.2 Fontes de Dados 35

4.3 Infraestrutura de implementação do data warehouse 36

5 AVALIAÇÃO EXPERIMENTAL - ETL E CONSULTAS ANALÍTICAS 37

5.1 ETL 37

5.1.1 Tabela de fatos 37

5.1.2 Tabelas de dimensão 38

5.2 Exemplo de Consulta Analítica 39

6 CONCLUSÕES 43

6.1 Resultados e Limitações 43

6.2 Trabalhos Futuros 43

Referências 45

APÊNDICES 47

APÊNDICE A – PACOTES E BIBLIOTECAS INSTALADAS NO AMBIENTE VIRTUAL DE DESENVOLVIMENTO 49

APÊNDICE B – *ETL - EXTRACT, TRANSFORM AND LOAD* . . 53

APÊNDICE C – EXEMPLOS DE CONSULTAS ANALÍTICAS . . . 93

ANEXOS 109

ANEXO A – BASES DE DADOS DO SUS FORNECIDAS PELO DESAFIO DATATHON 111

1 INTRODUÇÃO

O Departamento de Informática do Sistema Único de Saúde (DATASUS) disponibiliza informações relacionadas ao Sistema Único de Saúde (SUS) que podem servir para subsidiar análises objetivas da situação sanitária, tomadas de decisão baseadas em evidências e elaboração de programas de ações de saúde (DATASUS, 2023a). Essas informações, disponíveis para consulta e download na página web do DATASUS[1], estão organizadas em diferentes sistemas e bases de dados que são alimentados pelos serviços de saúde do SUS em todo o Brasil.

O presente trabalho tem interesse particular em um desses sistemas: o Sistema de Informações Ambulatoriais do SUS (SIA). As bases de dados públicas são anonimizadas para garantir a privacidade dos pacientes, mas contém atributos semi-identificadores, como idade (ou data de nascimento), sexo e município. Contém também atributos sensíveis, como os tipos de enfermidades tratadas e os tratamentos utilizados; e atributos não sensíveis, que são relevantes para análises e estudos, como os custos dos atendimentos e tratamentos para o SUS.

Considerando a abrangência nacional do SUS, a quantidade de dados armazenados nesse sistema é consideravelmente grande: apenas para o período de 2016 a 2020, há um total de cerca de 1,7 bilhões de registros, armazenados e disponibilizados em arquivos do sistema legado Tabwin. Esses mesmos 1,7 bilhões de registros ocupam cerca de 24GB em formato Apache Parquet, um formato mais moderno e eficiente para o armazenamento e recuperação de dados (PARQUET, 2023).

Mesmo disponibilizando os dados, o DATASUS não oferece uma solução eficiente para a consulta e análise dos dados do SUS, dificultando o trabalho de analistas e cientistas de dados interessados em resolver problemas relacionados à saúde pública no Brasil. De certo modo, isso vai contra a própria missão do DATASUS, que é "Promover modernização por meio da tecnologia da informação para apoiar o Sistema Único de Saúde – SUS" (DATASUS, 2023b).

O objetivo principal deste trabalho é extrair e limpar os dados da tabela SIA (e também das tabelas auxiliares), e fazer o carregamento desses dados em um data warehouse de arquitetura moderna e considerada estado da arte, que permite a realização de agrupamentos e análises avançadas da maneira mais eficiente possível.

[1] <https://datasus.saude.gov.br/transferencia-de-arquivos>

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os conceitos fundamentais relacionados a data warehouses tradicionais. Serão discutidos os princípios, arquiteturas tradicionais e funcionalidades essenciais dessa tecnologia. Também serão apresentados alguns conceitos básicos de Big Data e como essas tecnologias podem ser incorporados ao data warehousing.

2.1 O Data Warehouse Tradicional

O conceito de Data Warehouse (DW) é central para a gestão eficiente de dados em ambientes empresariais dos mais diversos.

De acordo com Inmon, Strauss and Neushloss (2010), o Data Warehouse é a base para o processamento de informação. Ele é definido por ser orientado a um assunto, integrado, não-volátil, variante em função do tempo e por ser uma coleção de dados que suporta a tomada de decisão da administração.

Elmasri Ramez; Navathe (2016, p. 1101) definem data warehouses como coleções de dados que armazenam e mantêm dados analíticos, separadamente das bases de dados transacionais, com o objetivo de apoiar a tomada de decisões. Os data warehouses geralmente mantêm dados de vários anos a fim de permitir a análise histórica dos dados, fornecendo armazenamento, funcionalidade e capacidade de respostas à consultas que vai além das capacidades de bases de dados transacionais.

Kimball and Ross (2013, pp. 3–4) definem os seguintes requisitos fundamentais de um Data Warehouse:

- O DW deve tornar as informações facilmente acessíveis;

- O DW deve apresentar as informações de maneira consistente;

- O DW deve se adaptar à mudanças;

- O DW deve apresentar as informações em tempo hábil;

- O DW deve ser um bastião seguro que protege as informações;

- O DW deve servir como a base confiável e de autoridade para uma melhor tomada de decisões;

- A comunidade de usuários deve aceitar e utilizar o DW para considerá-lo bem sucedido.

2.1.1 Arquitetura do Data Warehouse

As camadas de um data warehouse tradicional são mostradas na Figura 1, elaborada por Brito (2017), que também as descreve de maneira concisa na sua tese de doutorado, que é utilizada como referência para as próximas seções.

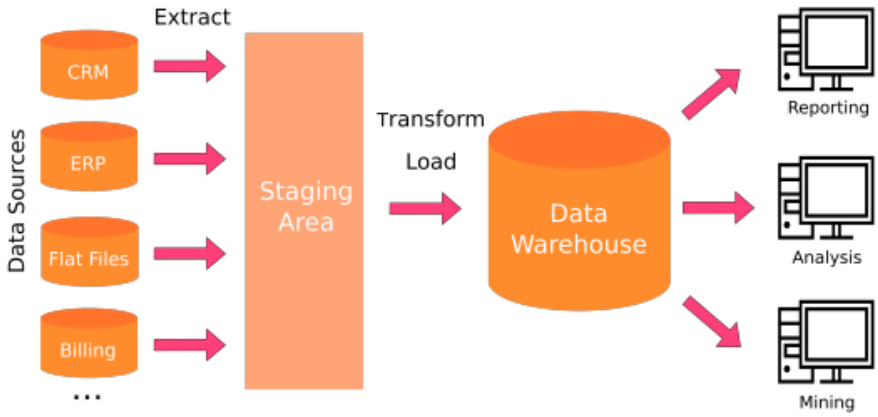

Figura 1 – Arquitetura de um data warehouse tradicional. (BRITO, 2017)

2.1.1.1 Fontes de Dados

Os dados que alimentam um DW vêm majoritariamente de sistemas transacionais, como CRMs (que são sistemas de *customer relationship management* ou gerenciamento de relacionamento com clientes), ERPs (*enterprise resource planning* ou planejamento de recursos da empresa), dados de faturamento, etc., que são utilizados para gerenciar o dia a dia operacional das organizações. Esses dados de diferentes fontes, geralmente heterogêneos, demandam processos de integração para se transformarem em informação confiável. Esses processos de integração são conhecidos como ETL (*Extract, Tranform and Load* ou Extração, Transformação e Carregamento).

2.1.1.2 *ETL - Extract, Load and Transform*

Os dados extraídos das fontes são enviados para uma área de preparação ou *staging area*, separada dos sistemas transacionais e também do data warehouse, onde são manipulados para garantir sua confiabilidade, consistência e para garantir a sua conformidade com os formatos de dados e esquemas de tabelas do Data Warehouse. Uma vez que os dados estão em conformidade, eles são finalmente carregados no DW. Os processos de ETL geralmente são executados em intervalos determinados de tempo para refletir as mudanças das bases de dados operacionais.

Kimball and Ross (2013, p. 499) afirmam que há uma multitude de ferramentas de ETL disponíveis no mercado e reforça que a utilização de uma ferramenta de ETL é considerada um padrão da indústria e uma melhor prática. Afirma, ainda, que as grandes

vantagens dessas ferramentas vêm com as fases futuras, particularmente com futuras modificações dos sistemas existentes.

2.1.1.3 Data Warehouse

Brito (2017) descreve o Data Warehouse como uma base de dados organizada especialmente para armazenar dados orientados a assuntos, integrados, históricos e não-voláteis. Essa base de dados foi abastecida com os dados provenientes dos processos integração da fase de ETL e estão prontos para permitir análises detalhadas, através de consultas analíticas, fornecendo informações estratégicas para auxiliar nos processos de tomada de decisão.

Os dados no DW são organizados de acordo com o modelo dimensional, que será melhor detalhado nas seções seguintes.

2.1.1.4 Aplicações de *Business Intelligence*

A última camada do DW é a de aplicações de Business Intelligence (BI), que são *softwares* utilizados pelas empresas para analisar dados e gerar conhecimento sobre o negócio, acessando os dados do Data Warehouse, geralmente através de *Structured Query Language* (em português, linguagem de consulta estruturada) ou, simplesmente, SQL. Essas aplicações são de diversas categorias como geração de relatórios, *dashboards* ou painéis de controle, mineração de dados, etc. (BRITO, 2017).

2.1.2 Modelagem Dimensional

De acordo com Kimball and Ross (2013, p. 7), a modelagem dimensional é uma técnica de longa data utilizada para tornar simples os bancos de dados. A modelagem dimensional é amplamente aceita como a técnica preferida para apresentar dados analíticos porque ela, simultaneamente, entrega dados que são compreensíveis pelos usuários de negócios e rapidez na realização das consultas.

Modelos dimensionais são chamados também de *star schemas*, ou esquemas estrela devido a sua semelhança com uma estrutura de estrela, como pode ser visto na Figura 2, elaborada por Kimball and Ross (2013).

Kimball and Ross (2013, p. 538) reforçam a importância de também pensar e modelar dimensionalmente os dados no contexto de Big Data. Os conceitos de Big Data serão explorados na seção 2.2.

O modelo dimensional possui dois elementos chave: as tabelas de fatos e as tabelas de dimensões.

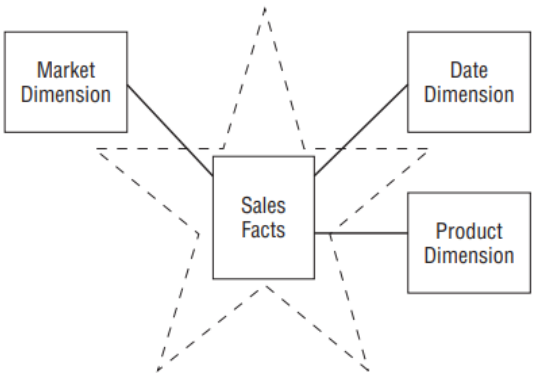

Figura 2 – *Star Schema* ou Modelo Dimensional. (KIMBALL; ROSS, 2013)

2.1.2.1 Tabelas de Fatos para Medições

Kimball and Ross (2013, p.10) definem as tabelas de fatos como tabelas que armazenam as medições resultantes dos eventos de processos de negócio de uma empresa. O termo fato representa uma medição ou evento de negócio, como por exemplo a passagem de um produto na caixa registradora de um supermercado durante uma venda. Nesse exemplo, cada linha ou registro de uma tabela de fatos deve representar um produto que passou pela caixa registradora. Os fatos mais úteis são numéricos e aditivos, como valores em moeda corrente e quantidades.

Tabelas de fatos, por representarem ocorrências de eventos de negócios, podem conter bilhões de registros.

2.1.2.2 Tabelas de Dimensão para Contexto Descritivo

Já as tabelas de dimensão, de acordo com Kimball and Ross (2013, p. 13), são companheiras integrais de uma tabela de fatos e contém o contexto textual associado aos eventos ou medições de processos. Elas descrevem o "quem, quê, onde, quando e o porquê"associados a um registro de uma tabela de fatos.

No exemplo de um produto passando por uma caixa registradora de supermercado, a tabela de dimensão de produto teria colunas como a descrição do produto, nome da marca, categoria do produto, nome do departamento, etc.

Normalmente, duas ou mais tabelas de dimensão estão associadas à uma tabela de fato. Tabelas de dimensão, geralmente, têm várias colunas ou atributos, além de terem menos registros que tabelas de fatos.

A Figura 3, retirada de Kimball and Ross (2013, p. 16), mostra uma junção da tabela de fatos com as tabelas de dimensão para um caso que poderia ser o exemplo da venda no supermercado.

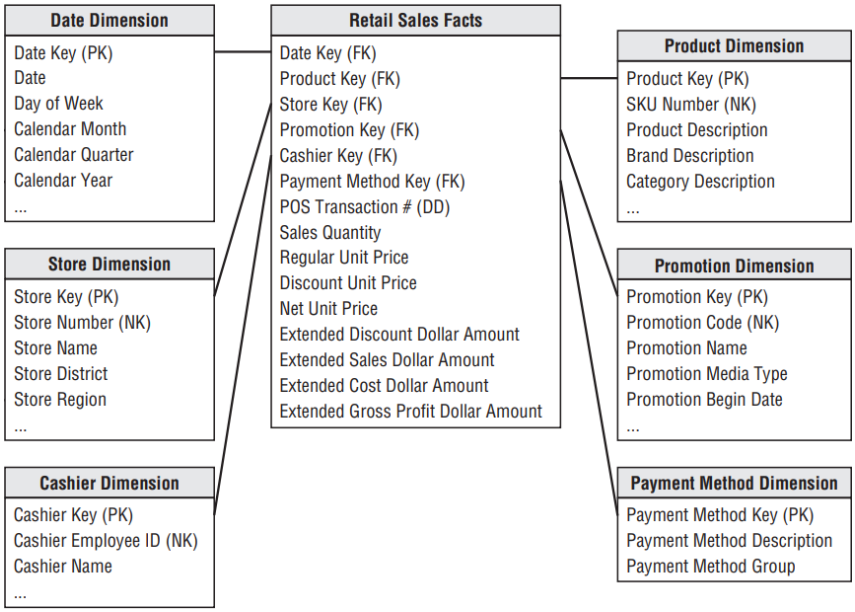

Figura 3 – Exemplo da junção da tabela de fatos com as tabelas de dimensão. (KIMBALL; ROSS, 2013)

2.2 Tecnologias de Big Data

Brito (2017, p. 50) afirma que embora o termo *Big Data* passe a impressão de ser relacionado apenas a grandes volumes de dados, essa noção pode induzir ao erro. Normalmente, a definição de Big Data é ligada aos "3 Vs": volume, variedade e velocidade. Ainda em 2012, cerca de 2,5 exabytes (1 exabyte = 1 bilhão de gigabytes) de dados já eram criados todos os dias e esse volume de dados duplicava a cada 40 meses aproximadamente (MCAFEE *et al.*, 2012, pp. 4–5).

É possível complementar a definição de Big Data com Kimball and Ross (2013, p. 527), que afirma que o grande volume não é sua característica mais interessante e que muitos dados considerados Big Data não podem ser analisados com nada que se pareça com SQL: "dados estruturados, semiestruturados, desestruturados e dados brutos em diferentes formatos, em alguns casos totalmente diferentes dos números escalares e textos armazenados em data warehouses nos últimos 30 anos."

Para lidar com os desafios apresentados pelo Big Data, diferentes tecnologias foram desenvolvidas. As mais relevantes para esse trabalho são abordadas nas seções seguintes.

2.2.1 MPP ou Processamento Paralelo Massivo

Brito (2017, p. 44) explica que o termo Processamento Paralelo Massivo (MPP, na sigla em inglês) refere-se ao uso coordenado de múltiplos processadores para executar uma tarefa em paralelo. Para fins de eficiência, MPPs geralmente são construídas de maneira que cada servidor no grupo (*cluster*, em inglês) roda em paralelo e de maneira

independente, com sua própria memória, armazenamento e processadores, compartilhando apenas a rede de comunicação. Nesse tipo de arquitetura, a ampliação das capacidades de processamento é alcançada através da adição de mais servidores ao *cluster*.

2.2.2 Armazenamento colunar e Apache Parquet

Em bancos de dados relacionais clássicos, as tuplas de uma tabela são armazenadas como uma sequência de linhas. Já no armazenamento colunar, ou orientado à colunas, cada coluna de uma tabela é gravada em disco contiguamente. Nesse tipo de armazenamento colunar, geralmente, cada atributo de uma tabela é gravado num arquivo separado e cada tupla é associada com uma chave única, que é usada para reconstruir as tuplas. Técnicas de compressão e armazenamento de metadados são utilizadas para melhorar a performance de armazenamento e consulta. O aspecto negativo do armazenamento colunar está relacionado ao custo de operações de atualização (*updates*), já que, geralmente, são divididas entre diversas colunas, armazenadas em diferentes arquivos (BRITO, 2017, p. 45).

O Apache Parquet é um formato *open-source* de arquivo de dados, orientado à colunas, que foi desenvolvido para oferecer armazenamento e recuperação eficiente dos dados. O Parquet fornece esquemas de compressão e codificação de alta performance para manipular dados complexos em grandes quantidades e é suportada em várias linguagens de programação e ferramentas de análise (PARQUET, 2023).

2.2.3 Apache Spark

O Apache Spark[1] é um motor de processamento multilinguagem para execução de engenharia de dados, ciência de dados e aprendizado de máquina, em máquinas de nó único ou em grupos (*clusters*) de computação.

O funcionamento do Spark se baseia em computação *in-memory* realizada através da sua abstração *Resilient Distributed Dataset* (conjunto de dados distribuído e resiliente, em português). RDDs são estruturas de dados paralelas e tolerantes à falhas que permitem a persistência de resultados intermediários em disco. Uma aplicação Spark é gerenciada por um programa *driver*, responsável por alocar recursos computacionais com a ajuda de um gerenciador de *cluster*. Todas as operações nos RDDs são primeiramente mapeadas em um *Directed Acyclic Graph* (DAG) ou grafo acíclico dirigido, em português. Em seguida, as operações são reorganizadas pelo agendador do DAG em grupos de tarefas menores baseados nas suas dependências comuns. Essas tarefas são executadas por processos chamados executores que rodam nos nós de trabalho do *cluster* (BRITO, 2017).

Quando executado em um nó único, como no caso da avaliação experimental desse trabalho, o Spark é capaz de paralelizar e distribuir as tarefas para todos os núcleos lógicos do processador local.

[1] <https://spark.apache.org/>

3 TRABALHOS RELACIONADOS

A criação de data warehouses para dados públicos de saúde ou a integração de dados de diferentes bases do SUS para a criação de DWs é tema de diversos artigos e trabalhos acadêmicos já produzidos. Com o aumento de popularidade e interesse em *big data* e tecnologias de processamento de grandes volumes de dados, trabalhos utilizando Apache Spark para manipular dados de saúde pública também surgiram. Nesse capítulo, alguns desses trabalhos são explorados e suas semelhanças e diferenças, discutidas.

Freire, Souza and Almeida (2015) desenvolveram um data warehouse através da integração de três diferentes sistemas de informação do SUS: Sistema de Informações Hospitalares (SIH), o módulo de oncologia do sistema de Autorização de Procedimentos Ambulatoriais de Alta Complexidade (APAC-ONCO) e o Sistema de Informação sobre Mortalidade (SIM) para o estado do Rio de Janeiro, com dados do período de janeiro de 2000 a dezembro de 2004. As três tabelas, somando 5,6 milhões de registros, foram carregadas no sistema de banco de dados relacional MySQL Server para a realização do processo de ETL e o data warehouse foi implementado com o MySQL e a ferramenta de *business intelligence* Pentaho. Nesse trabalho, os autores tiveram acesso a dados não-anonimizados, o que os permitiu executar a integração das três bases das dados através de métodos de pareamento de registros *record linkage*.

Pinto (2016), na sua dissertação de mestrado, utilizou técnicas de ETL para desenvolver um módulo de pré-processamento e pareamento probabilístico de registros utilizando Apache Spark. Para o seu trabalho, focado em avaliar o impacto do Programa Bolsa Família sobre as doenças infecciosas tuberculose e hanseníase, foram utilizadas bases de dados do Cadastro Unico (CadÚnico), Sistema de Informação de Agravos de Notificação (SINAN), Sistema de Informações Hospitalares (SIH), Sistema de Informações de Mortalidade (SIM) e das Folhas de Pagamento do Bolsa Família (PBF). As quatro base de dados englobam o período de 1998 a 2013 e, combinadas, possuem pouco mais de 115 milhões de registros. A autora também teve acesso a informação não-anonimizadas, como nome completo do paciente, número identificação social (NIS), registro geral (RG) e cadastro de pessoa física (CPF). A solução de pré-processamento foi implementada em um *cluster* de 8 processadores.

Barreto *et al.* (2019) descrevem a criação do Centro de Integração de Dados e Conhecimentos para Saúde (CIDACS) em 2016, em Salvador, na Bahia, como parte da Fundação Oswaldo Cruz (FIOCRUZ), instituição afiliada ao Ministério da Saúde do Brasil. O CIDACS conta com diversas bases de dados de sistemas governamentais: CadÚnico, PBF, Minha Casa Minha Vida, Programa de Fornecimento de Sisternas Públicas, SIM, Sistema de Informação sobre Nascidos Vivos (SINASC), Sistema de Informação de Agravos

de Notificação (SINAN), SIH, e alguns outros, totalizando cerca de 350 milhões de registros. Por se tratar de instituição governamental e ter acesso à informações não-anonimizadas, também realiza processos de pareamento de registro durante os processos de ETL. No fluxo de ingestão de dados do data warehouse CIDACS, são utilizados arquivos do tipo Parquet e o Apache Spark é a ferramenta usada para pré e pós processamento dos dados, como pode ser visto na Figura 4.

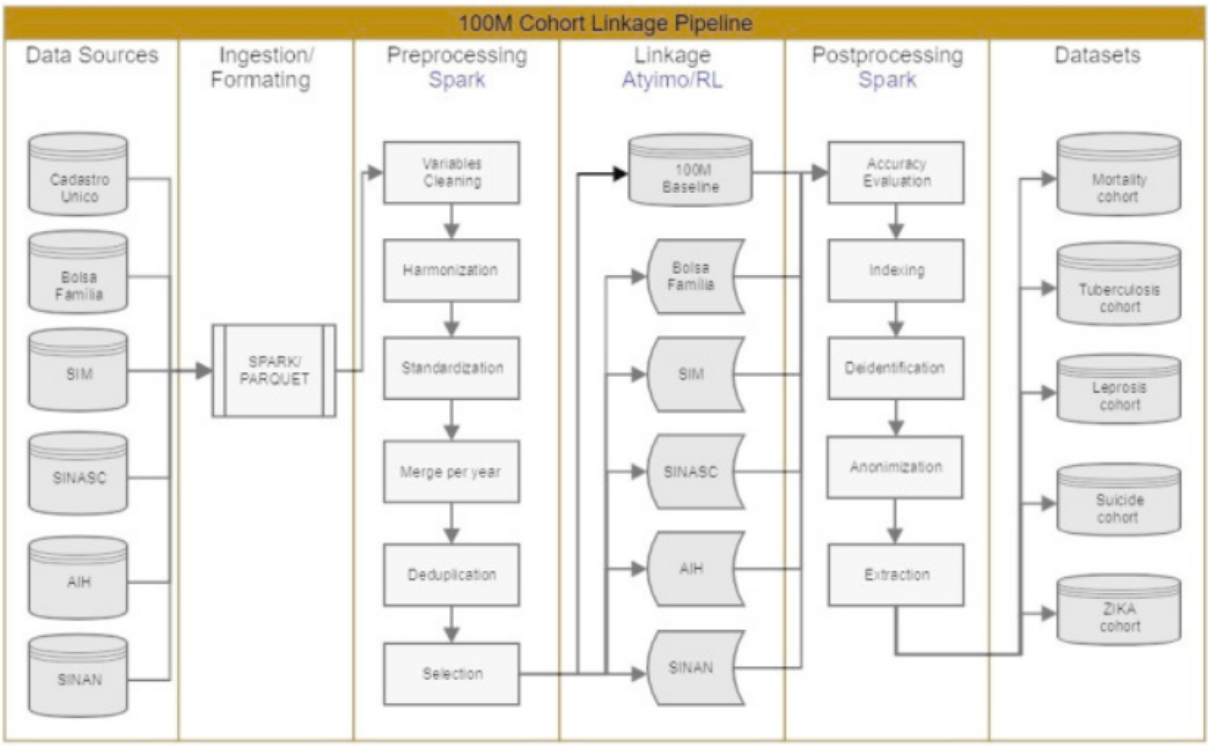

Figura 4 – Fluxo de ingestão de dados do CIDACS. (BARRETO *et al.*, 2019)

Ao contrário dos trabalhos apresentados nesse capítulo, o presente trabalho não explorou a aplicação de métodos de pareamento e/ou deduplicação de registros, uma vez que os dados utilizados nesse trabalho são dados públicos e anonimizados do SUS, disponíveis para a população em geral, que não contam com nenhum atributo identificador.

4 PROPOSTA - FONTES DE DADOS E INFRAESTRUTURA

Esse capítulo apresenta o Sistema de Informações Ambulatoriais (SIA) e descreve as fontes de dados utilizadas para a criação do data warehouse para dados ambulatoriais do SUS, e também a infraestrutura de *hardware* e *software* usada para a implementação do DW.

4.1 Sistema de Informações Ambulatoriais (SIA)

O SIA foi implementado no Brasil em 1995 e registra atendimentos ambulatoriais de pacientes através de Boletins de Produção Ambulatorial (BPA) e Autorizações de Procedimento de Alta Complexidade (APAC). O processamento dos dados ocorre de maneira decentralizada, onde cada estado e município pode registar, programar, processar e cobrar pela produção ambulatorial das instalações de saúde sob a sua administração. O SIA é amplamente utilizado para estudos de Avaliação de Tecnologias em Saúde, já que fornece a quantidade de procedimentos realizados e o custo desses procedimentos para o SUS (ALI *et al.*, 2019; JUNIOR *et al.*, 2018).

A base de dados SIA está disponível para download na página de transferência de arquivos do DATASUS[1], juntamente com outras bases de dados de sistemas de saúde do SUS. Além dos dados principais do SIA, também são disponibilizados arquivos auxiliares para tabulação no formato de arquivos de definição do Tabwin, programa de tabulação desenvolvido pelo próprio DATASUS, e documentação explicativa. No entanto, as bases de dados são disponibilizadas para *download* no formato .dbc, que são arquivos .dbf (dBase) comprimidos e criptografados pelo algoritmo privado PKWare (MENDES, 2019). Esse formato de arquivo é uma barreira para o fácil acesso e manipulação dos dados, principalmente no contexto de *big data*.

4.2 Fontes de Dados

Nesse trabalho, para contornar a necessidade de baixar um volume massivo de dados em arquivos no formato .dbc do DATASUS, foi utilizado um conjunto de dados fornecidos pela equipe de engenheira de dados do Hospital Israelita Albert Einstein, na ocasião da competição "Desafio Datathon – O impacto da Dermatite Atópica na saúde pública"[2], organizada pela Eretz.bio, ecossistema de inovação do Hospital Israelita Albert Einstein, que aconteceu no segundo semestre de 2021. Essas bases de dados foram disponibilizadas no formato Apache Parquet e não sofreram qualquer tipo de pré-processamento, apenas

[1] <https://datasus.saude.gov.br/transferencia-de-arquivos>
[2] <https://www.eretz.bio/desafio-datathon/>

passaram pela conversão de formato. O documento que descreve as bases de dados fornecidas pelo Desafio Datathon, de autoria da própria equipe do desafio, se encontra no Anexo A. A Tabela 1 mostra as bases de dados desse conjunto que foram utilizadas nesse trabalho.

Base	**Descrição**	**Atributos**	**Registros**	**Tamanho**
SIA	Sistema de Informações Ambulatoriais	60	1.742.743.969	23,6 GB
CNES	Cadastro Nacional de Estabelecimentos de Saúde	54	418.045	62,0 MB
SIGTAP	Sistema de Gerenciamento da Tabela de Procedimentos	16	275.577	1,46 MB
CID	Código Internacional de Doenças	6	14.230	296 KB

Tabela 1 – Bases de dados do Desafio Datathon utilizadas nesse trabalho. Elaboração do autor.

Além do conjunto de dados do Desafio Datathon, para a construção de algumas tabelas de dimensão, foram utilizados os arquivos de conversão de Tabwin, disponibilizados pelo DATASUS como arquivos auxiliares para tabulação da base de dados SIA. Esses arquivos, no entanto, são exponencialmente menores do que as bases de dados da Tabela 1, possuindo poucas dezenas de registros no máximo. Os detalhes da construção dessas tabelas de dimensão serão mostrados no Capítulo 5.

4.3 Infraestrutura de implementação do data warehouse

Para a execução desse trabalho, foi utilizado um *laptop* Lenovo S-145, com processador AMD Ryzen 7 3700U 2.30 GHz, de 4 núcleos físicos e 8 núcleos lógicos de processamento, placa de vídeo *onboard* Radeon Vega 10 Mobile Gfx, 8 GB de memória RAM total, com 2 GB de RAM reservados para a memória de vídeo, armazenamento SSD de 256 GB e sistema operacional Windows 11 de 64 bits.

O motor de processamento utilizado para os processos de ETL dos dados e para as consultas analíticas no data warehouse foi o Apache Spark 3.5.1. Foi criado um ambiente virtual de desenvolvimento com Python versão 3.10.5, onde foi instalado a biblioteca Jupyter Notebook, para a implementação dos códigos de execução. Nesse ambiente virtual, também foi instalado a biblitoteca PySpark, que faz a interface entre Python e o Apache Spark. A lista completa de bibliotecas e módulos de Python (e suas versões) instalados pode ser encontrada no Apêndice A.

No contexto desse trabalho, o Spark foi utilizado em um único *laptop* e não em um *cluster* de computação distribuída. O Spark é capaz de paralelizar as tarefas de execução, utilizando todos os núcleos lógicos de processamento locais ao mesmo tempo.

5 AVALIAÇÃO EXPERIMENTAL - ETL E CONSULTAS ANALÍTICAS

A execução desse trabalho foi implementada em dois *Jupyter Notebooks*: no *notebook* do Apêndice B, foi executado o processo de ETL (extração, transformação e carregamento) das fontes de dados com PySpark. No *notebook* do Apêndice C, a tabela de fatos e as tabelas de dimensões foram carregadas e, como exemplo das capacidades do DW, algumas consultas analíticas foram realizadas.

Nesse capítulo, será detalhado o processo de ETL, o modelo dimensional dos dados no DW e será mostrado um exemplo de consulta analítica das inúmeras possíveis.

5.1 ETL

O processo de ETL gerou uma tabela de fatos e doze tabelas de dimensão conforme a Tabela 2, todas armazenadas em formato Apache Parquet. É possível observar que a tabela de fatos SIA tem, de longe, o maior número de registros: pouco mais de 1,7 bilhões. A segunda e terceira maiores tabelas, respectivamente, são as tabelas de dimensão CNES (Cadastro Nacional de Estabelecimentos de Saúde, com 418 mil registros) e a SIGTAP_PROCED (Sistema de Gerenciamento da Tabela de Procedimentos, com 275 mil registros). Em um contexto de junção da tabela de fatos com essas duas tabelas de dimensão em um *cluster* de computação distribuída, o Spark é capaz de utilizar a técnica de *broadcasting*, que consiste em enviar os dados de tabelas que cabem na memória RAM para todos os nós de processamento do *cluster*, economizando assim comunicação dentro rede e ganhando em performance de consulta (BRITO, 2017). As demais tabelas de dimensão, exponencialmente menores, também se beneficiam dessa técnica.

5.1.1 Tabela de fatos

A tabela de fatos SIA bruta, como pode ser baixada do DATASUS, possui 60 atributos/colunas. Vários desses atributos são campos lógicos (ou booleanos), com valores 0 ou 1, ou são atributos que não adicionam valor aos dados do ponto de vista de paciente, patologias, enfermidades ou custos para o SUS. Para a execução desse trabalho, esses campos foram eliminados. Das 60 colunas originais, apenas 19 foras selecionados para armazenamento no DW. A relação dos atributos e descrição de cada um está no *snippet* de código mostrada na Figura 5.

Dos 19 atributos selecionados, apenas dois são somáveis - podem passar por operações de soma quando agrupados: `PA_QTAPR` (quantidade de procedimentos aprovados) e `PA_VALAPR` (valor aprovado de procedimentos). O atributo `PA_IDADE` (idade do paciente) pode passar por operação de média aritmética, mas não pode ser somado.

Base	Descrição	Fato ou Dimensão	Atributos	Registros	Tamanho
SIA	Sistema de Informações Ambulatoriais	Fato	19	1.742.743.969	15,5 GB
CNES	Cadastro Nacional de Estabelecimentos de Saúde	Dimensão	56	418.045	62,0 MB
SIGTAP_PROCED	Sistema de Gerenciamento da Tabela de Procedimentos	Dimensão	16	275.577	1,46 MB
CID	Código Internacional de Doenças	Dimensão	8	14.230	296 KB
MUNICÍPIOS	Listagem do IBGE de todos os municípios brasileiros, estados, regiões e outras informações	Dimensão	10	5.570	147 KB
CBOCOD	Código Brasileiro de Ocupações	Dimensão	2	2.812	63 KB
ANO_MES	Data no formato AAAAMM	Dimensão	7	72	2,66 KB
TPUPS	Tipos de Estabelcimentos de Saúde	Dimensão	2	42	7,55 KB
MOTSAI	Motivos de saída	Dimensão	2	22	7,5 KB
CATEND	Caráter de Atendimento	Dimensão	2	18	7,32 KB
RACA_COR	Raça/Cor do paciente	Dimensão	2	14	6,65 KB
DOCORIG	Tipo de Documento de Origem da produção ambulatorial	Dimensão	2	6	5,58 KB
SEXO	Sexo do paciente	Dimensão	2	3	2,64 KB

Tabela 2 – Tabelas de fatos e dimensões. Elaboração do autor.

5.1.2 Tabelas de dimensão

As tabelas de dimensão CNES, SIGTAP e CID foram carregadas a partir dos conjuntos de dados disponibilizados pelo Desafio Datathon, conforme anexo A.

A tabela de dimensão MUNICIPIOS foi construída pelo autor a partir de dados baixados diretamente do Instituto Brasileiro de Geografia e Estatísticas (IBGE)[1]. A tabela ANO_MES também foi criada pelo autor, adaptando Kimball and Ross (2013, pp. 79–83).

As demais tabelas de dimensão (CBOCOD, TPUPS, MOTSAI, CATEND, RACA_COR, DOCORIG e SEXO) foram construídas a partir dos arquivos auxiliares de tabulação disponibilizados pelo DATASUS[2].

[1] <https://servicodados.ibge.gov.br/api/docs/localidades>
[2] <https://datasus.saude.gov.br/transferencia-de-arquivos/>

```
[8]: # Seleciona as colunas de interesse da tabela SIA_PA

sia_df = sia_df \
    .selectExpr('PA_CMP',  # Data da Realização do Procedimento / Competência␣
 ↪(AAAAMM)
                'PA_CODUNI',  # Código do SCNES do estabelecimento de saúde
                'PA_TPUPS',  # Tipo do estabelecimento
                'PA_UFMUN',  # Município onde está localizado o estabelecimento
                'PA_PROC_ID',  # Código do procedimento ambulatorial
                'PA_DOCORIG',  # Instrumento de registro: C: BPA-C, I: BPA-I, P:
 ↪ APAC-P, S: APAC-S, A: RAAS-AD, B: RAAS-Psico
                'PA_CNSMED',  # Número CNS do profissional de saúde executante
                'PA_CBOCOD',  # Código de ocupação brasileira do profissional
                'PA_MOTSAI',  # Motivo de saída ou zeros, caso não tenha
                'PA_CIDPRI',  # CID principal (APAC ou BPA-I)
                'PA_CIDSEC',  # CID secundário (APAC)
                'PA_CIDCAS',  # CID causas associadas (APAC)
                'PA_CATEND',  # Caráter de Atendimento (APAC ou BPA-I)
                'cast(PA_IDADE as int)',  # Idade do paciente em anos
                'PA_SEXO',  # Sexo do paciente
                'PA_RACACOR',  # Raça/cor do paciente
                'PA_MUNPCN',  # Município de residência do paciente ou do␣
 ↪estabelecimento, caso não se tenha à identificação do paciente o que ocorre␣
 ↪no (BPA)  # BPA-C
                'cast(PA_QTDAPR as int)',  # Quantidade aprovado do procedimento
                'cast(PA_VALAPR as float)')  # Valor aprovado do procedimento
```

Figura 5 – Seleção dos atributos de interesse da tabela SIA. Elaboração do autor.

5.2 Exemplo de Consulta Analítica

Com a tabela de fatos e as tabelas de dimensão devidamente armazenadas no formato Apache Parquet e organizadas de acordo com o modelo dimensional mostrado na Figura 6, o DW está pronto para receber consultas analíticas. Para esse trabalho, os arquivos Parquet resultantes dos processos de ETL foram carregados no Jupyter Notebook mostrado no Apêndice C, para receberem consultas através da API (Interface de Programação da Aplicação) do PySpark.

Na consulta exemplo mostrada da Figura 7, um hipotético usuário do DW busca responder a também hipotética questão: *"Qual é a quantidade de procedimentos e valores aprovados pelo SUS para procedimentos Psicoterápicos, durantes os anos de 2018 a 2020, realizados em Unidades Básicas de Saúde, em pacientes Pretos e do sexo Masculino, agrupados por Região do País, Ano e Categoria do CID Primário?"*

Essa consulta faz a junção da tabela de fatos com 7 diferentes tabelas de dimensão e estabelece condições de filtragem em 5 tabelas de dimensão. Também é realizado um agrupamento de atributos de 3 tabelas de dimensão diferentes, realizando a soma de 2 atributos da tabela de fatos e ordenando o resultado por 3 atributos diferentes.

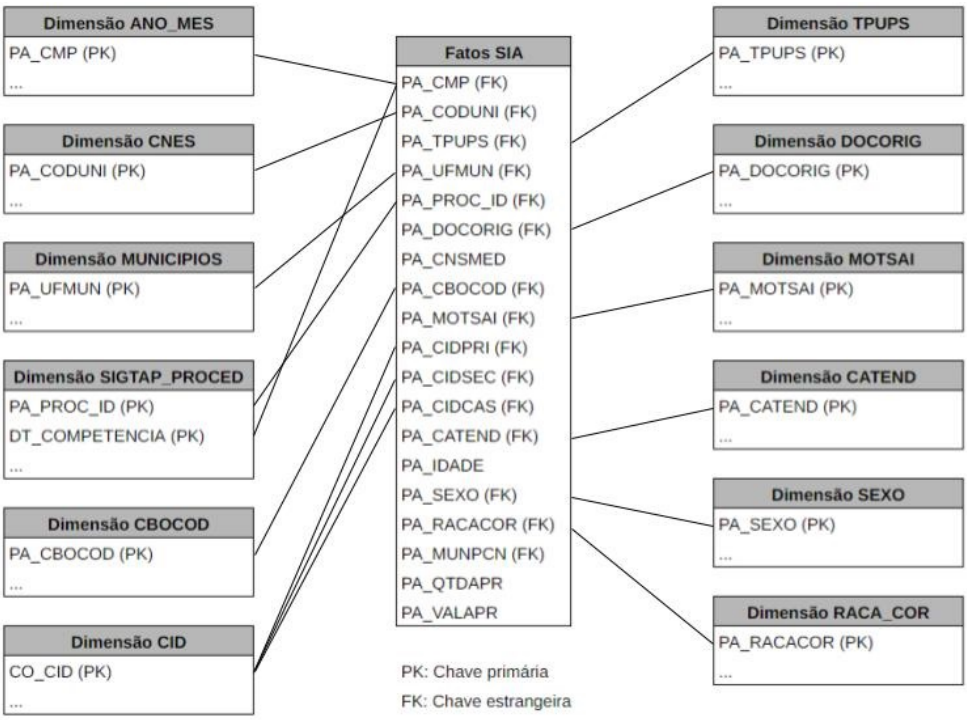

Figura 6 – Diagrama do modelo dimensional do DW. Apenas chaves primárias sendo mostradas nas tabelas de dimensão. Elaboração do autor.

Essa consulta, quando executada pelo mesmo *laptop* descrito na Seção 4.3, retorna o resultado em cerca de 5,7 minutos, após ler 7,8 GB de dados e pelo menos 1,7 bilhões de registros dos arquivos Parquet, como pode ser visto na Figura 8. O resultado retornado por essa consulta pode ser encontrado no notebook do Apêndice C.

```
[10]: sia_df \
    .join(municipios_df.select('PA_UFMUN', 'REGIAO_SIGLA'), on='PA_UFMUN') \
    .join(ano_mes_df.select('PA_CMP', 'ANO'), on='PA_CMP') \
    .join(cnes_df.select('PA_CODUNI', 'TP_UNIDADE_DESC'), on='PA_CODUNI') \
    .join(sigtap_proced_df.select('PA_PROC_ID', 'NO_PROCEDIMENTO',␣
  ↪'DT_COMPETENCIA'), on=[sia_df['PA_PROC_ID'] ==␣
  ↪sigtap_proced_df['PA_PROC_ID'],

  ↪          sia_df['PA_CMP'] == sigtap_proced_df['DT_COMPETENCIA']]) \
    .join(cid_df.select('CO_CID', 'NO_CID', 'CO_CATEG', 'NO_CATEG'),␣
  ↪on=sia_df['PA_CIDPRI'] == cid_df['CO_CID']) \
    .join(sexo_df, on='PA_SEXO') \
    .join(raca_cor_df, on='PA_RACACOR') \
    .where('TP_UNIDADE_DESC LIKE "%UNIDADE BASICA%"') \
    .where('NO_PROCEDIMENTO LIKE "%PSICOTERAPIA%"') \
    .where('ANO in ("2018", "2019", "2020")') \
    .where('SEXO_DESC = "Masculino"') \
    .where('RACACOR_DESC = "PRETA"') \
    .select('REGIAO_SIGLA', 'ANO', 'CO_CATEG', 'NO_CATEG', 'PA_QTDAPR',␣
  ↪'PA_VALAPR' ) \
    .groupBy('REGIAO_SIGLA', 'ANO', 'CO_CATEG', 'NO_CATEG' ) \
    .agg({'PA_QTDAPR':'sum', 'PA_VALAPR':'sum'}) \
    .sort('REGIAO_SIGLA', 'ANO', 'sum(PA_VALAPR)', ascending=[True, True,␣
  ↪False]) \
    .withColumn('sum(PA_VALAPR)', F.format_number('sum(PA_VALAPR)', 2)) \
    .withColumnRenamed('sum(PA_QTDAPR)', 'PROCEDIMENTOS') \
    .withColumnRenamed('sum(PA_VALAPR)', 'VALOR') \
    .show(60, truncate=False)
```

Figura 7 – Exemplo de consulta analítica utilizando o modelo dimensional. Elaboração do autor.

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
23	showString at NativeMethodAccessorImpl.java:0 +details	2024/10/15 19:16:07	5,7 min	164/164	7.8 GiB			28.0 KiB

Figura 8 – Tempo de execução da consulta de exemplo de acordo com a interface de usuário do Apache Spark. Elaboração do Autor.

6 CONCLUSÕES

6.1 Resultados e Limitações

Esse trabalho foi bem sucedido no seu objetivo de manipular um grande volume de dados - pouco mais de 1,7 bilhões de registros - provenientes da produção ambulatorial do SUS, que compreendem os últimos meses de 2015 até o final de 2020. Foi capaz de organizá-los em um modelo dimensional de acordo com as melhores práticas apontadas por (KIMBALL; ROSS, 2013), através da utilização de duas das mais populares ferramentas do atual panorama de tecnologias para *big data*: o formato de arquivos colunares Apache Parquet, para o armazenamento e consulta eficiente de dados, e o motor de processamento paralelizado e distribuído de dados Apache Spark.

Mesmo utilizando dados públicos anonimizados e disponíveis para o público geral, a criação desse data warehouse permite consultas abrangentes a partir do ponto de vista de atendimentos únicos, estabelecimentos de saúde, procedimentos realizados, doenças e as outras dimensões disponíveis no DW. Permite ainda responder onde, quando e quanto custou determinada doença ou procedimento para o SUS. No entanto, a anonimidade dos dados e a falta de um atributo identificador único por indivíduo impedem análises a partir do ponto de vista (ou granularidade) de paciente. Essa é uma questão que não buscou ser resolvida nesse trabalho, uma vez que outros projetos e autores, citados no Capítulo 3, tiveram acesso à informações identificadores dos pacientes.

A maior limitação desse trabalho está na camada de análise de dados (ou *business intelligence*) do data warehouse: para realizar consultas analíticas no DW, respondendo perguntas e gerando conhecimento sobre a produção ambulatorial do SUS, é necessário que o analista ou usuário domine a API (interface de programação) do PySpark. Ou seja, é necessário conhecer os diversos comandos que permitem realizar operações e opções de consultas dos dados.

6.2 Trabalhos Futuros

O passo seguinte natural para o trabalho executado nesse projeto de data warehouse seria melhorar a camada de *business intelligence* (BI) ou consultas analíticas. A integração de uma ferramenta de BI com o DW, como Tableau ou Power BI, permitiria que analistas interessados em extrair conhecimento dos dados tivessem condições de acessar DW de uma maneira mais amigável, sem a necessidade de ter conhecimento avançado na API do PySpark.

Uma vez que as etapas de ETL foram definidas nesse trabalho, é possível também carregar o DW com os dados mais recentes do Sistema de Informações ambulatoriais do SUS

(SIA), a partir do ano de 2021. Nesse caso, uma etapa adicional de ETL seria necessária: a conversão dos dados do formato .dbf - formato no qual os dados são disponibilizados no DATASUS - para o formato Parquet.

Além dos dados mais recentes da tabela SIA, seria possível integrar nesse DW outros sistemas de informação do SUS, como o Sistema de Informações Hospitalares (SIH) e o Sistema de Informações sobre Mortalidade (SIM), dentre outros. Esses diferentes sistemas de informação se beneficiaram de algumas das tabelas de dimensão criadas para o data warehouse.

Finalmente, outro possível trabalho futuro seria a implementação desse data warehouse em um *cluster* de computação distribuída. A performance do Spark executado num único *laptop* de configurações modestas se mostrou eficiente e suficiente para manipular um enorme volume de dados. A mudança para um ambiente de processamento distribuído, com *hardware* profissional, poderia extrair ainda mais performance das capacidades do Apache Spark.

REFERÊNCIAS

ALI, M. S. *et al.* Administrative data linkage in brazil: potentials for health technologycassessment. **Frontiers in pharmacology**, Frontiers Media SA, v. 10, p. 984, 2019.

BARRETO, M. L. *et al.* The centre for data and knowledge integration for health (cidacs): linking health and social data in brazil. **International journal of population data science**, Swansea University, v. 4, n. 2, 2019.

BRITO, J. J. **Data Warehouses na era do Big Data: processamento eficiente de Junções Estrela no Hadoop**. 2017. Tese (Doutorado) — Universidade de São Paulo, 2017.

DATASUS. **Acesso à Informação**. 2023. Disponível em: <https://datasus.saude.gov.br/acesso-a-informacao/>. Acesso em 19 março 2023.

DATASUS. **Sobre o DATASUS**. 2023. Disponível em: <https://datasus.saude.gov.br/sobre-o-datasus/>. Acesso em 19 março 2023.

ELMASRI RAMEZ; NAVATHE, S. B. **Fundamentals of database systems seventh edition**. Hoboken: Pearson, 2016.

FREIRE, S. M.; SOUZA, R. C. d.; ALMEIDA, R. T. d. Integrating brazilian health information systems in order to support the building of data warehouses. **Research on Biomedical Engineering**, SciELO Brasil, v. 31, n. 3, p. 196–207, 2015.

INMON, W. H.; STRAUSS, D.; NEUSHLOSS, G. **DW 2.0: The architecture for the next generation of data warehousing**. [*S.l.: s.n.*]: Elsevier, 2010.

JUNIOR, A. A. G. *et al.* Building the national database of health centred on the individual: administrative and epidemiological record linkage-brazil, 2000-2015. **International Journal of Population Data Science**, Swansea University, v. 3, n. 1, 2018.

KIMBALL, R.; ROSS, M. **The data warehouse toolkit: the complete guide to dimensional modeling**. [*S.l.: s.n.*]: John Wiley & Sons, 2013.

MCAFEE, A. *et al.* Big data: the management revolution. **Harvard business review**, Cambridge, v. 90, n. 10, p. 60–68, 2012.

MENDES, D. P. Ferramenta de pré-processamento e visualização de dados do datasus. UEMA, 2019.

PARQUET. **Apache Parquet: Motivation**. 2023. Disponível em: <https://parquet.apache.org/docs/overview/motivation/>. Acesso em 13 novembro 2023.

PINTO, C. d. S. Aplicação de etl para a integração de dados com ênfase em big data na área de saúde pública. Instituto de Matemática. Departamento de Ciência da Computação, 2016.

APÊNDICES

APÊNDICE A – PACOTES E BIBLIOTECAS INSTALADAS NO AMBIENTE VIRTUAL DE DESENVOLVIMENTO

Lista de pacotes e bibliotecas Python gerada através do comando ‘pip freeze‘ na janela do Terminal, com o ambiente virtual ativado.

anyio==4.4.0
argon2-cffi==23.1.0
argon2-cffi-bindings==21.2.0
arrow==1.3.0
asttokens==2.4.1
async-lru==2.0.4
attrs==23.2.0
Babel==2.15.0
beautifulsoup4==4.12.3
bleach==6.1.0
certifi==2024.7.4
cffi==1.16.0
charset-normalizer==3.3.2
colorama==0.4.6
comm==0.2.2
dbfread==2.0.7
debugpy==1.8.2
decorator==5.1.1
defusedxml==0.7.1
exceptiongroup==1.2.2
executing==2.0.1
fastjsonschema==2.20.0
findspark==2.0.1
fqdn==1.5.1
h11==0.14.0
httpcore==1.0.5
httpx==0.27.0
idna==3.7
ipykernel==6.29.5
ipython==8.26.0
ipywidgets==8.1.3
isoduration==20.11.0
jedi==0.19.1
Jinja2==3.1.4
json5==0.9.25
jsonpointer==3.0.0
jsonschema==4.23.0
jsonschema-specifications==2023.12.1
jupyter==1.0.0
jupyter-console==6.6.3
jupyter-events==0.10.0
jupyter-lsp==2.2.5
jupyter_client==8.6.2
jupyter_core==5.7.2
jupyter_server==2.14.2
jupyter_server_terminals==0.5.3
jupyterlab==4.2.4
jupyterlab_pygments==0.3.0
jupyterlab_server==2.27.3
jupyterlab_widgets==3.0.11
MarkupSafe==2.1.5
matplotlib-inline==0.1.7
mistune==3.0.2
nbclient==0.10.0
nbconvert==7.16.4
nbformat==5.10.4
nest-asyncio==1.6.0
notebook==7.2.1
notebook_shim==0.2.4
overrides==7.7.0

packaging==24.1
pandocfilters==1.5.1
parso==0.8.4
platformdirs==4.2.2
prometheus_client==0.20.0
prompt_toolkit==3.0.47
psutil==6.0.0
pure_eval==0.2.3
py4j==0.10.9.7
pycparser==2.22
Pygments==2.18.0
python-dateutil==2.9.0.post0
python-json-logger==2.0.7
pywin32==306
pywinpty==2.0.13
PyYAML==6.0.1
pyzmq==26.0.3
qtconsole==5.5.2
QtPy==2.4.1
referencing==0.35.1
requests==2.32.3
rfc3339-validator==0.1.4
rfc3986-validator==0.1.1
rpds-py==0.19.1
Send2Trash==1.8.3
six==1.16.0
sniffio==1.3.1
soupsieve==2.5
stack-data==0.6.3
terminado==0.18.1
tinycss2==1.3.0
tomli==2.0.1
tornado==6.4.1
traitlets==5.14.3
types-python-dateutil==2.9.0.20240316
typing_extensions==4.12.2
uri-template==1.3.0
urllib3==2.2.2
wcwidth==0.2.13
webcolors==24.6.0
webencodings==0.5.1
websocket-client==1.8.0
wget==3.2
widgetsnbextension==4.0.11

APÊNDICE B – *ETL - EXTRACT, TRANSFORM AND LOAD*

O Jupyter Notebook `sus_dw_etl.ipynb` pode ser baixado em <https://github.com/DaniloGouvea/sus-dw/blob/main/sus_dw_etl.ipynb>.

sus_dw_etl

October 8, 2024

1 Criação de data warehouse para dados públicos de atendimentos ambulatoriais do SUS

Esse Jupyter notebook é parte do trabalho de conclusão de curso do MBA em Inteligência Artificial e Big Data, oferecido pelo ICMC - USP, do aluno Danilo Gouvea Silva, da Turma 3.

Parte do capítulo de Avaliação Experimental da monografia, nesse primeiro notebook (`sus_dw_etl.ipynb`), está a execução do processo de ETL - "Extract, Transform and Load" - dos dados brutos de saúde provenientes dos bancos de dados públicos do DATASUS. Após o processo de ETL, o data warehouse estará organizado em 1 tabela de fatos e 12 tabelas de dimensões, todas armazendas em arquivos Apache Parquet.

No segundo notebook (`sus_dw_eda.ipynb`), todas as tabelas serão carregadas e estarão prontras para a análise exploratória através de consultas analíticas.

Esse notebook foi criado e utilizado localmente. Para utilizá-lo, é necessário que estejam localmente instalados o Spark, o Java e o Python. Também é recomendado a criação de um ambiente virtual Python para a instalação de todos pacotes de Python necessários, que estão contidos no arquivo de requisitos `requirements.txt`, que disponibilizado junto a esse notebook.

É importante ressaltar que o objetivo desse notebook não é demonstrar, nem guiar a instalação e configuração do Spark numa máquina local.

1.1 Criação da sessão Spark

Nessa seção, é criado uma sessão local de Spark, com apenas um nó mestre e sem nós de trabalho e gerenciador de cluster. As tarefas (tasks) serão executadas pelo driver localizado no nó mestre e utilizarão o máximo de núcleos lógicos de processamento disponíveis no processador local.

[1]:
```
# Verifica as versões instaladas de Java, Python e Spark

!java -version
!python --version
!pyspark --version
```

```
java version "1.8.0_411"
Java(TM) SE Runtime Environment (build 1.8.0_411-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.411-b09, mixed mode)

Python 3.10.5
```

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.1
      /_/

Using Scala version 2.12.18, Java HotSpot(TM) 64-Bit Server VM, 1.8.0_381
Branch HEAD
Compiled by user heartsavior on 2024-02-15T11:24:58Z
Revision fd86f85e181fc2dc0f50a096855acf83a6cc5d9c
Url https://github.com/apache/spark
Type --help for more information.
```

[2]:
```
# Configura corretamente as variáveis de ambiente do Spark

!pip install findspark
import findspark
findspark.init()
```

```
Requirement already satisfied: findspark in c:\mba\tcc\venv\lib\site-packages
(2.0.1)
```

[3]:
```
# Cria a sessão de Spark

from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName('sus_dw_etl') \
    .master('local[*]') \
    .getOrCreate()

spark
```

[3]:
```
<pyspark.sql.session.SparkSession at 0x215574bd690>
```

1.2 Carregamento da tabela de fatos SIASUS - SERVIÇO DE INFORMAÇÕES AMBULATORIAIS DO SUS (`sia_df`)

[4]:
```
# Importa as funções e tipos de dados do Spark

import pyspark.sql.functions as F
import pyspark.sql.types as T
```

[5]:
```
# Carrega num dataframe os arquivos da tabela SIA - Atendimentos Ambulatoriais

sia_df = spark.read.option('mergeSchema', 'true').parquet('data/tb_sia_pa')
```

[6]:
```
# Conta o número total de registros

sia_df_count = sia_df.count()

print(f'Número de registros: {sia_df_count:,}')
```

```
Número de registros: 1,742,743,969
```

[7]:
```
# Exibe a lista de colunas do dataframe SIA

sia_df \
    .select('*')
```

[7]:
```
DataFrame[PA_CODUNI: string, PA_GESTAO: string, PA_CONDIC: string, PA_UFMUN:
string, PA_REGCT: string, PA_INCOUT: string, PA_INCURG: string, PA_TPUPS:
string, PA_TIPPRE: string, PA_MN_IND: string, PA_CNPJCPF: string, PA_CNPJMNT:
string, PA_CNPJ_CC: string, PA_MVM: string, PA_CMP: string, PA_PROC_ID: string,
PA_TPFIN: string, PA_SUBFIN: string, PA_NIVCPL: string, PA_DOCORIG: string,
PA_AUTORIZ: string, PA_CNSMED: string, PA_CBOCOD: string, PA_MOTSAI: string,
PA_OBITO: string, PA_ENCERR: string, PA_PERMAN: string, PA_ALTA: string,
PA_TRANSF: string, PA_CIDPRI: string, PA_CIDSEC: string, PA_CIDCAS: string,
PA_CATEND: string, PA_IDADE: string, IDADEMIN: string, IDADEMAX: string,
PA_FLIDADE: string, PA_SEXO: string, PA_RACACOR: string, PA_MUNPCN: string,
PA_QTDPRO: string, PA_QTDAPR: string, PA_VALPRO: string, PA_VALAPR: string,
PA_UFDIF: string, PA_MNDIF: string, PA_DIF_VAL: string, NU_VPA_TOT: string,
NU_PA_TOT: string, PA_INDICA: string, PA_CODOCO: string, PA_FLQT: string,
PA_FLER: string, PA_ETNIA: string, PA_VL_CF: string, PA_VL_CL: string,
PA_VL_INC: string, PA_SRV_C: string, PA_INE: string, PA_NAT_JUR: string]
```

[8]:
```
# Seleciona as colunas de interesse da tabela SIA_PA

sia_df = sia_df \
    .selectExpr('PA_CMP',  # Data da Realização do Procedimento / Competência␣
↪(AAAAMM)
                'PA_CODUNI',  # Código do SCNES do estabelecimento de saúde
                'PA_TPUPS',  # Tipo do estabelecimento
                'PA_UFMUN',  # Município onde está localizado o estabelecimento
                'PA_PROC_ID',  # Código do procedimento ambulatorial
                'PA_DOCORIG',  # Instrumento de registro: C: BPA-C, I: BPA-I, P:
↪ APAC-P, S: APAC-S, A: RAAS-AD, B: RAAS-Psico
                'PA_CNSMED',  # Número CNS do profissional de saúde executante
                'PA_CBOCOD',  # Código de ocupação brasileira do profissional
                'PA_MOTSAI',  # Motivo de saída ou zeros, caso não tenha
                'PA_CIDPRI',  # CID principal (APAC ou BPA-I)
                'PA_CIDSEC',  # CID secundário (APAC)
                'PA_CIDCAS',  # CID causas associadas (APAC)
                'PA_CATEND',  # Caráter de Atendimento (APAC ou BPA-I)
                'cast(PA_IDADE as int)',  # Idade do paciente em anos
```

```
                'PA_SEXO',  # Sexo do paciente
                'PA_RACACOR',  # Raça/cor do paciente
                'PA_MUNPCN',  # Município de residência do paciente ou do␣
 ↪estabelecimento, caso não se tenha à identificação do paciente o que ocorre␣
 ↪no (BPA)  # BPA-C
                'cast(PA_QTDAPR as int)',  # Quantidade aprovado do procedimento
                'cast(PA_VALAPR as float)')  # Valor aprovado do procedimento
```

[9]:
```
# Escreve a tabela de fatos `sia_df` em arquivo parquet no diretório␣
 ↪sus-data-warehouse

sia_df.write \
    .format('parquet') \
    .mode('overwrite') \
    .save('sus-data-warehouse/sia')
```

1.2.1 Breve análise exploratória

Análise exploratória de algumas atributos (dimensões) da tabela de fatos que chamaram atenção do autor.

[10]:
```
# Mostra o primeiro registro do data frame 'sia_df'

sia_df.show(1, vertical=True)
```

```
-RECORD 0---------------------
 PA_CMP     | 201907
 PA_CODUNI  | 0003786
 PA_TPUPS   | 07
 PA_UFMUN   | 292740
 PA_PROC_ID | 0417010060
 PA_DOCORIG | I
 PA_CNSMED  | 108258251530008
 PA_CBOCOD  | 225310
 PA_MOTSAI  | 00
 PA_CIDPRI  | 0000
 PA_CIDSEC  | 0000
 PA_CIDCAS  | 0000
 PA_CATEND  | 01
 PA_IDADE   | 60
 PA_SEXO    | F
 PA_RACACOR | 99
 PA_MUNPCN  | 292740
 PA_QTDAPR  | 1
 PA_VALAPR  | 15.15
only showing top 1 row
```

[11]:
```
sia_df \
    .groupBy('PA_MUNPCN') \
    .count() \
    .orderBy(F.desc('count')) \
    .withColumn('MUNPCN_PERCENT', F.round(F.col('count') / sia_df_count * 100,␣
 ↪2)) \
    .withColumn('count', F.format_number('count', 0)) \
    .show(5)
```

```
+---------+-----------+--------------+
|PA_MUNPCN|      count|MUNPCN_PERCENT|
+---------+-----------+--------------+
|   999999|460,061,443|          26.4|
|   330455|103,426,318|          5.93|
|   355030| 70,764,436|          4.06|
|   310620| 58,555,360|          3.36|
|   431490| 29,844,593|          1.71|
+---------+-----------+--------------+
only showing top 5 rows
```

[12]:
```
# Quantidade de registros com PA_UFMUN (Município do estabelecimento) <>␣
 ↪PA_MUNPCN (Município do paciente) e que não são BPC-Consolidado
# (BPC-C não possui informação de paciente)

sia_df \
    .select('PA_UFMUN', 'PA_MUNPCN') \
    .where('PA_UFMUN <> PA_MUNPCN and PA_MUNPCN <> "999999"') \
    .count()
```

[12]: 309678509

Quantidade de registros com PA_MUNPCN == '999999' é igual a quantidade de registros com PA_DOCORIG == 'BPA - Consolidado'. Após consulta analítica, é possível afirmar que todos esses registros tem PA_DOCORIG == PA_DOCORIG == 'BPA - Consolidado'

[13]:
```
# Verifica a cardinalidade de cada coluna

sia_df \
    .agg({c: 'approx_count_distinct' for c in sia_df.columns}) \
    .show(vertical=True)
```

```
-RECORD 0----------------------------------
 approx_count_distinct(PA_CODUNI)  | 88345
 approx_count_distinct(PA_QTDAPR)  | 26993
 approx_count_distinct(PA_DOCORIG) | 6
 approx_count_distinct(PA_UFMUN)   | 5522
 approx_count_distinct(PA_SEXO)    | 6
 approx_count_distinct(PA_MUNPCN)  | 5637
```

```
approx_count_distinct(PA_CIDCAS)  | 3225
approx_count_distinct(PA_CMP)     | 67
approx_count_distinct(PA_CIDPRI)  | 13852
approx_count_distinct(PA_MOTSAI)  | 21
approx_count_distinct(PA_CNSMED)  | 492359
approx_count_distinct(PA_CIDSEC)  | 2321
approx_count_distinct(PA_CATEND)  | 24
approx_count_distinct(PA_VALAPR)  | 256163
approx_count_distinct(PA_TPUPS)   | 41
approx_count_distinct(PA_RACACOR) | 25
approx_count_distinct(PA_CBOCOD)  | 219
approx_count_distinct(PA_PROC_ID) | 2864
approx_count_distinct(PA_IDADE)   | 129
```

[14]:
```
# Conta valores faltantes em todas as colunas

sia_df  \
    .select([F.count(F.when(F.col(c).isNull() | \
                            F.isnan(c), c)
                    ).alias(c) \
             for c in sia_df.columns]) \
    .show(vertical=True)
```

```
-RECORD 0--------------
 PA_CMP     | 0
 PA_CODUNI  | 0
 PA_TPUPS   | 0
 PA_UFMUN   | 0
 PA_PROC_ID | 0
 PA_DOCORIG | 0
 PA_CNSMED  | 0
 PA_CBOCOD  | 809921
 PA_MOTSAI  | 0
 PA_CIDPRI  | 0
 PA_CIDSEC  | 0
 PA_CIDCAS  | 49825202
 PA_CATEND  | 0
 PA_IDADE   | 0
 PA_SEXO    | 0
 PA_RACACOR | 0
 PA_MUNPCN  | 0
 PA_QTDAPR  | 0
 PA_VALAPR  | 0
```

[15]:
```
# Verifica a contagem de valores na coluna PA_CATEND
```

```
sia_df \
    .groupBy('PA_CATEND') \
    .count() \
    .orderBy(F.desc('count')) \
    .withColumn('count', F.format_number('count', 0)) \
    .show()
```

```
+---------+-------------+
|PA_CATEND|        count|
+---------+-------------+
|       01|1,100,224,673|
|       99|  460,061,443|
|       02|  180,461,010|
|       06|    1,167,119|
|       03|      555,353|
|       05|      172,030|
|       04|      102,214|
|       00|          107|
|      056|            2|
|      026|            2|
|      046|            2|
|      027|            2|
|      065|            2|
|      030|            1|
|      022|            1|
|      070|            1|
|      067|            1|
|      057|            1|
|      044|            1|
|      086|            1|
+---------+-------------+
only showing top 20 rows
```

Coluna PA_ETNIA foi removida da tabela de fatos `sia_df` após análise dos valores presentes. Cardinalidade muito grande, valores não explicáveis e 99,9% dos registros 'NULL'.

[17]:
```
# Verifica quantidade de CIDs de acordo com números de dígitos do CID

sia_df \
    .selectExpr('len(PA_CIDPRI) as LEN') \
    .groupBy('LEN') \
    .count() \
    .show()
```

```
+---+----------+
|LEN|     count|
+---+----------+
|  3|  48835518|
```

```
|  4|1693908411|
|  2|        31|
|  1|         9|
+---+----------+
```

[18]:
```
# Mostra CIDs com quantidade de dígitos igual a 2 ou menores

sia_df \
    .selectExpr('len(PA_CIDPRI) as LEN_CIDPRI',
                'PA_CIDPRI') \
    .where('LEN_CIDPRI <= 2') \
    .show()
```

```
+----------+---------+
|LEN_CIDPRI|PA_CIDPRI|
+----------+---------+
|         2|       N8|
|         2|       H0|
|         1|        S|
|         1|        G|
|         2|       F2|
|         2|       H3|
|         2|       R9|
|         2|       R9|
|         1|        Z|
|         1|        Z|
|         2|       H6|
|         2|       I7|
|         2|       R0|
|         1|        Z|
|         1|        Z|
|         2|       B4|
|         2|       Q2|
|         2|       H5|
|         2|       H3|
|         2|       L1|
+----------+---------+
only showing top 20 rows
```

1.3 Carregamento das tabelas de dimensões

Nessa seção, serão carregadas as tabelas de dimensões: - `municipios_df`: listagem do IBGE de todos os munícipios brasileiros, estados, regiões e outras informações; - `cnes_df`: Cadastro Nacional de Estabelecimentos de Saúde; - `sigtap_proced_df`: listagem dos Procedimentos oferecidos pelo SUS; - `cid_df`: CID-10 Código Internacional de Doenças; - `ano_mes_df`: Dimensão "data"no formato AAAAMM; - `cbocod_df`: Código Brasileiro de Ocupações; - `tpups_df`: Tipos de Esta-

belcimentos de Saúde; - `catend_df`: Caráter de Atendimento; - `docorig_df`: Tipo de Documento de Origem da produção ambulatorial; - `sexo_df`: Sexo do paciente; - `raca_cor_df`: Raça/Cor do paciente; - `mosai_df`: Motivos de saída

1.3.1 Dimensão MUNICIPIOS (`municipios_df`) (`sia_df['PA_UFMUN']`)

[19]:
```
# Importa o módulo requests para baixar os d

import requests
```

[20]:
```
# Baixa os dados de municípios do IBGE

municipios = requests.get('https://servicodados.ibge.gov.br/api/v1/localidades/
↪municipios').json()
```

[21]:
```
# Mostra o formato do primeiro município

municipios[0]
```

[21]:
```
{'id': 1100015,
 'nome': "Alta Floresta D'Oeste",
 'microrregiao': {'id': 11006,
  'nome': 'Cacoal',
  'mesorregiao': {'id': 1102,
   'nome': 'Leste Rondoniense',
   'UF': {'id': 11,
    'sigla': 'RO',
    'nome': 'Rondônia',
    'regiao': {'id': 1, 'sigla': 'N', 'nome': 'Norte'}}}},
 'regiao-imediata': {'id': 110005,
  'nome': 'Cacoal',
  'regiao-intermediaria': {'id': 1102,
   'nome': 'Ji-Paraná',
   'UF': {'id': 11,
    'sigla': 'RO',
    'nome': 'Rondônia',
    'regiao': {'id': 1, 'sigla': 'N', 'nome': 'Norte'}}}}}
```

[22]:
```
# Cria o dataframe de municípios

municipios_df = spark.createDataFrame(
    data=[(str(municipio['id'])[:6],  # 'Slicing para remover o dígito␣
↪verificador, o sétimo dígito.
           municipio['nome'],
           municipio['microrregiao']['nome'],
           municipio['microrregiao']['mesorregiao']['nome'],
           municipio['microrregiao']['mesorregiao']['UF']['sigla'],
```

```
            municipio['microrregiao']['mesorregiao']['UF']['nome'],
            municipio['microrregiao']['mesorregiao']['UF']['regiao']['sigla'],
            municipio['microrregiao']['mesorregiao']['UF']['regiao']['nome'],
            municipio['regiao-imediata']['nome'],
            municipio['regiao-imediata']['regiao-intermediaria']['nome']) for␣
 ↪municipio in municipios],
    schema=['PA_UFMUN', 'NOME', 'MICRORREGIAO', 'MESORREGIAO', 'UF', 'UF_NOME',␣
 ↪'REGIAO_SIGLA', 'REGIAO', 'REGIAO_IMEDIATA', 'REGIAO_INTERMEDIARIA']
)

municipios_df.show()
```

```
+--------+--------------------+-----------------+-----------------+---+--------
+------------+------+---------------+--------------------+
|PA_UFMUN|                NOME|     MICRORREGIAO|      MESORREGIAO| UF|
UF_NOME|REGIAO_SIGLA|REGIAO|REGIAO_IMEDIATA|REGIAO_INTERMEDIARIA|
+--------+--------------------+-----------------+-----------------+---+--------
+------------+------+---------------+--------------------+
|  110001|Alta Floresta D'O…|           Cacoal|Leste Rondoniense| RO|Rondônia|
N| Norte|         Cacoal|           Ji-Paraná|
|  110002|           Ariquemes|        Ariquemes|Leste Rondoniense| RO|Rondônia|
N| Norte|      Ariquemes|         Porto Velho|
|  110003|             Cabixi|Colorado do Oeste|Leste Rondoniense| RO|Rondônia|
N| Norte|        Vilhena|           Ji-Paraná|
|  110004|             Cacoal|           Cacoal|Leste Rondoniense| RO|Rondônia|
N| Norte|         Cacoal|           Ji-Paraná|
|  110005|          Cerejeiras|Colorado do Oeste|Leste Rondoniense| RO|Rondônia|
N| Norte|        Vilhena|           Ji-Paraná|
|  110006|   Colorado do Oeste|Colorado do Oeste|Leste Rondoniense| RO|Rondônia|
N| Norte|        Vilhena|           Ji-Paraná|
|  110007|          Corumbiara|Colorado do Oeste|Leste Rondoniense| RO|Rondônia|
N| Norte|        Vilhena|           Ji-Paraná|
|  110008|       Costa Marques|    Guajará-Mirim|  Madeira-Guaporé| RO|Rondônia|
N| Norte|      Ji-Paraná|           Ji-Paraná|
|  110009|     Espigão D'Oeste|           Cacoal|Leste Rondoniense| RO|Rondônia|
N| Norte|         Cacoal|           Ji-Paraná|
|  110010|       Guajará-Mirim|    Guajará-Mirim|  Madeira-Guaporé| RO|Rondônia|
N| Norte|    Porto Velho|         Porto Velho|
|  110011|               Jaru|        Ji-Paraná|Leste Rondoniense| RO|Rondônia|
N| Norte|           Jaru|         Porto Velho|
|  110012|           Ji-Paraná|        Ji-Paraná|Leste Rondoniense| RO|Rondônia|
N| Norte|      Ji-Paraná|           Ji-Paraná|
|  110013|  Machadinho D'Oeste|        Ariquemes|Leste Rondoniense| RO|Rondônia|
N| Norte|           Jaru|         Porto Velho|
|  110014|Nova Brasilândia …| Alvorada D'Oeste|Leste Rondoniense| RO|Rondônia|
N| Norte|         Cacoal|           Ji-Paraná|
|  110015| Ouro Preto do Oeste|        Ji-Paraná|Leste Rondoniense| RO|Rondônia|
```

```
N| Norte|      Ji-Paraná|           Ji-Paraná|
|  110018|       Pimenta Bueno|          Vilhena|Leste Rondoniense| RO|Rondônia|
N| Norte|         Cacoal|           Ji-Paraná|
|  110020|         Porto Velho|      Porto Velho|  Madeira-Guaporé| RO|Rondônia|
N| Norte|    Porto Velho|         Porto Velho|
|  110025|   Presidente Médici|        Ji-Paraná|Leste Rondoniense| RO|Rondônia|
N| Norte|      Ji-Paraná|           Ji-Paraná|
|  110026|          Rio Crespo|        Ariquemes|Leste Rondoniense| RO|Rondônia|
N| Norte|      Ariquemes|         Porto Velho|
|  110028|      Rolim de Moura|           Cacoal|Leste Rondoniense| RO|Rondônia|
N| Norte|         Cacoal|           Ji-Paraná|
+--------+------------------+----------------+----------------+---+--------
+------------+------+--------------+--------------------+
only showing top 20 rows
```

[23]:
```
# Conta a quantidade total de registros do data frame de munícipios

municipios_df.count()
```

[23]: 5570

[24]:
```
# Salva a tabela `municipios_df` na pasta do data warehouse␣
↪`sus-data-warehouse` em formato Apache Parquet

municipios_df.write \
    .format('parquet') \
    .mode('overwrite') \
    .save('sus-data-warehouse/municipios')
```

1.3.2 Dimensão CADASTRO NACIONAL DE ESTABELECIMENTOS DE SAÚDE (cnes_df) (sia_df['PA_CODUNI'])

[25]:
```
# Carrega num dataframe os arquivos da tabela CNES - Cadastro Nacional de␣
↪Estabelecimentos de Saúde

cnes_df = spark.read.option('mergeSchema', 'true').parquet('data/
↪tb_cnes_estabelecimentos')
```

[26]:
```
# Mostra o primeiro registro do data frame CNES}

cnes_df.show(1, vertical=True, truncate=False)
```

```
-RECORD 0-----------------------------------------------
 CO_UNIDADE                  | 3556205173388
 CO_CNES                     | 5173388
 NU_CNPJ_MANTENEDORA         | NULL
 TP_PFPJ                     | 1
```

NIVEL_DEP	1
NO_RAZAO_SOCIAL	SILVIO MAURICIO RINHEL VIRDES
NO_FANTASIA	SILVIO MAURICIO RINHEL VIRDES
NO_LOGRADOURO	RUA 13 DE MAIO
NU_ENDERECO	133
NO_COMPLEMENTO	1 ANDAR
NO_BAIRRO	CENTRO
CO_CEP	13270020
CO_REGIAO_SAUDE	NULL
CO_MICRO_REGIAO	NULL
CO_DISTRITO_SANITARIO	NULL
CO_DISTRITO_ADMINISTRATIVO	NULL
NU_TELEFONE	19-38713106
NU_FAX	NULL
NO_EMAIL	NULL
NU_CPF	02926823878
NU_CNPJ	NULL
CO_ATIVIDADE	04
CO_CLIENTELA	03
NU_ALVARA	000266-1-1
DT_EXPEDICAO	03-out-2006 00:00:00
TP_ORGAO_EXPEDIDOR	2
DT_VAL_LIC_SANI	NULL
TP_LIC_SANI	NULL
TP_UNIDADE	22
CO_TURNO_ATENDIMENTO	03
CO_ESTADO_GESTOR	35
CO_MUNICIPIO_GESTOR	355620
DT_ATUALIZACAO	31/08/2019
CO_USUARIO	SANDRA
CO_CPFDIRETORCLN	02926823878
REG_DIRETORCLN	28264
ST_ADESAO_FILANTROP	NULL
CO_MOTIVO_DESAB	NULL
NO_URL	NULL
NU_LATITUDE	-22.9722215
NU_LONGITUDE	-46.9953355
DT_ATU_GEO	31/08/2019
NO_USUARIO_GEO	SANDRA
CO_NATUREZA_JUR	4000
TP_ESTAB_SEMPRE_ABERTO	N
ST_GERACREDITO_GERENTE_SGIF	NULL
ST_CONEXAO_INTERNET	S
CO_TIPO_UNIDADE	NULL
NO_FANTASIA_ABREV	NULL
TP_GESTAO	M
DT_ATUALIZACAO_ORIGEM	16/12/2006
CO_TIPO_ESTABELECIMENTO	016

```
 CO_ATIVIDADE_PRINCIPAL      | 001
 ST_CONTRATO_FORMALIZADO     | N
only showing top 1 row
```

[27]:
```
# Número de registros no data frame 'cnes_df'

cnes_df_count = cnes_df.count()
cnes_df_count
```

[27]: 418045

Dimensão CLASSIFICAÇÃO DE ESTABELECIMENTOS (tp_estab_df)

[28]:
```
# Cria o data frame da dimensão Classificação de Estabelecimentos␣
 ↪(CO_TIPO_ESTABELECIMENTO)

# CO_TIPO_ESTABELECIMENTO extraído da página http://cnes2.datasus.gov.br/, aba␣
 ↪Relatórios, opção Classif. de Estabelecimentos em 15/08/2024
tp_estab_df = spark.createDataFrame(
    data=[
        ('000', 'OUTROS'),
        ('001', 'UNIDADE BASICA DE SAUDE'),
        ('002', 'CENTRAL DE GESTAO EM SAUDE'),
        ('003', 'CENTRAL DE REGULACAO'),
        ('004', 'CENTRAL DE ABASTECIMENTO'),
        ('005', 'CENTRAL DE TRANSPLANTE'),
        ('006', 'HOSPITAL'),
        ('007', 'CENTRO DE ASSISTENCIA OBSTETRICA E NEONATAL NORMAL'),
        ('008', 'PRONTO ATENDIMENTO'),
        ('009', 'FARMACIA'),
        ('010', 'UNIDADE DE ATENCAO HEMATOLOGICA E/OU HEMOTERAPICA'),
        ('011', 'NUCLEO DE TELESSAUDE'),
        ('012', 'UNIDADE DE ATENCAO DOMICILIAR'),
        ('013', 'POLO DE PREVENCAO DE DOENCAS E AGRAVOS E PROMOCAO DA SAUDE'),
        ('014', 'CASAS DE APOIO A SAUDE'),
        ('015', 'UNIDADE DE REABILITACAO'),
        ('016', 'AMBULATORIO'),
        ('017', 'UNIDADE DE ATENCAO PSICOSSOCIAL'),
        ('018', 'UNIDADE DE APOIO DIAGNOSTICO'),
        ('019', 'UNIDADE DE TERAPIAS ESPECIAIS'),
        ('020', 'LABORATORIO DE PROTESE DENTARIA'),
        ('021', 'UNIDADE DE VIGILANCIA DE ZOONOSES'),
        ('022', 'LABORATORIO DE SAUDE PUBLICA'),
        ('023', 'CENTRO DE REFERENCIA EM SAUDE DO TRABALHADOR'),
        ('024', 'SERVICO DE VERIFICACAO DE OBITO'),
        ('025', 'CENTRO DE IMUNIZACAO')
    ],
```

```
    schema=['CO_TIPO_ESTABELECIMENTO', 'CO_TIPO_ESTABELECIMENTO_DESC']
)

tp_estab_df.show(truncate=False)
```

```
+----------------------+-------------------------------------------------------
---+
|CO_TIPO_ESTABELECIMENTO|CO_TIPO_ESTABELECIMENTO_DESC
|
+----------------------+-------------------------------------------------------
---+
|000                    |OUTROS
|
|001                    |UNIDADE BASICA DE SAUDE
|
|002                    |CENTRAL DE GESTAO EM SAUDE
|
|003                    |CENTRAL DE REGULACAO
|
|004                    |CENTRAL DE ABASTECIMENTO
|
|005                    |CENTRAL DE TRANSPLANTE
|
|006                    |HOSPITAL
|
|007                    |CENTRO DE ASSISTENCIA OBSTETRICA E NEONATAL NORMAL
|
|008                    |PRONTO ATENDIMENTO
|
|009                    |FARMACIA
|
|010                    |UNIDADE DE ATENCAO HEMATOLOGICA E/OU HEMOTERAPICA
|
|011                    |NUCLEO DE TELESSAUDE
|
|012                    |UNIDADE DE ATENCAO DOMICILIAR
|
|013                    |POLO DE PREVENCAO DE DOENCAS E AGRAVOS E PROMOCAO DA
SAUDE|
|014                    |CASAS DE APOIO A SAUDE
|
|015                    |UNIDADE DE REABILITACAO
|
|016                    |AMBULATORIO
|
|017                    |UNIDADE DE ATENCAO PSICOSSOCIAL
|
|018                    |UNIDADE DE APOIO DIAGNOSTICO
```

```
|
|019                    |UNIDADE DE TERAPIAS ESPECIAIS
|
+----------------------+-----------------------------------------------------
---+
only showing top 20 rows
```

Dimensão TIPO DE ESTABELECIMENTO (tp_unidade_df)

[29]:
```
# Cria o data frame da dimensão Tipo de Estabelecimento (TP_UNIDADE)
# CNES.TP_UNIDADE == SIA.PA_TPUPS

# Tipos de Estabelecimentos (TP_UNIDADE) extraídos do arquivo de conversão para␣
 ↪Tabwin TP_ESTAB.CNV
# baixado na página https://datasus.saude.gov.br/transferencia-de-arquivos/ em␣
 ↪15/08/2024
# Fonte: CNES - Cadastro Nacional de Estabelecimentos de Saúde, Modalidade:␣
 ↪Arquivos auxiliares para tabulação, Tipo de Arquivo: Arquivo de definição do␣
 ↪Tabwin
# /TAB_CNES/CNV/TP_ESTAB.CNV


## Importa a tabela TP_ESTAB.CNV e cria data frame
import csv

# Cria a lista para as tuplas
tp_unidade = []

# Exrai a tabela TP_ESTAB.CNV e coloca numa lista de tuplas
with open('data/TP_ESTAB.CNV', mode='r') as file:
    reader = csv.reader(file, delimiter='\t')
    # Pula a primeira linha do arquivo CNV
    next(reader)
    # Lê linha a linha (linhas com mais de um char), faz o slice e limpa os␣
 ↪espaços do início e fim de cada string resultante
    for row in reader:
        if len(row[0]) > 1:
            tp_unidade.append((row[0][112:114].strip(), row[0][11:112].strip()))

# Cria o data frame
tp_unidade_df = spark.createDataFrame(
    data=tp_unidade,
    schema=['TP_UNIDADE', 'TP_UNIDADE_DESC']
)

tp_unidade_df.show(truncate=False)
```

```
+----------+-------------------------------------------------------------+
|TP_UNIDADE|TP_UNIDADE_DESC                                              |
+----------+-------------------------------------------------------------+
|01        |POSTO DE SAUDE                                               |
|02        |CENTRO DE SAUDE/UNIDADE BASICA                               |
|04        |POLICLINICA                                                  |
|05        |HOSPITAL GERAL                                               |
|07        |HOSPITAL ESPECIALIZADO                                       |
|09        |PRONTO SOCORRO DE HOSPITAL GERAL (ANTIGO)                    |
|12        |PRONTO SOCORRO TRAUMATO-ORTOPEDICO (ANTIGO)                  |
|15        |UNIDADE MISTA                                                |
|20        |PRONTO SOCORRO GERAL                                         |
|21        |PRONTO SOCORRO ESPECIALIZADO                                 |
|22        |CONSULTORIO ISOLADO                                          |
|32        |UNIDADE MOVEL FLUVIAL                                        |
|36        |CLINICA/CENTRO DE ESPECIALIDADE                              |
|39        |UNIDADE DE APOIO DIAGNOSE E TERAPIA (SADT ISOLADO)           |
|40        |UNIDADE MOVEL TERRESTRE                                      |
|42        |UNIDADE MOVEL DE NIVEL PRE-HOSPITALAR NA AREA DE URGENCIA    |
|43        |FARMACIA                                                     |
|45        |UNIDADE DE SAUDE DA FAMILIA                                  |
|50        |UNIDADE DE VIGILANCIA EM SAUDE                               |
|60        |COOPERATIVA OU EMPRESA DE CESSAO DE TRABALHADORES NA SAUDE    |
+----------+-------------------------------------------------------------+
only showing top 20 rows
```

[30]:
```
# Completa o data frame 'cnes_df' com as descrições dos campos TP_UNIDADE e␣
↪CO_TIPO_ESTABELECIMENTO

cnes_df = cnes_df \
    .join(tp_unidade_df, on='TP_UNIDADE', how='left') \
    .join(tp_estab_df, on='CO_TIPO_ESTABELECIMENTO', how='left')
```

[31]:
```
# Verifica se todos os registros do data frame original cnes_df continuam após␣
↪as junções

cnes_df.count() == cnes_df_count
```

[31]:
```
True
```

[32]:
```
# Renomeia a coluna CO_UNIDADE para PA_CODUNI para padronizar com a tabela de␣
↪fatos SIA_PA

cnes_df = cnes_df \
    .withColumnRenamed('CO_CNES', 'PA_CODUNI')
```

[33]:
```
cnes_df.show(1, vertical=True, truncate=False)
```

```
-RECORD 0-----------------------------------------------------------------
 CO_TIPO_ESTABELECIMENTO      | NULL
 TP_UNIDADE                   | 36
 CO_UNIDADE                   | 3550307917376
 PA_CODUNI                    | 7917376
 NU_CNPJ_MANTENEDORA          | NULL
 TP_PFPJ                      | 3
 NIVEL_DEP                    | 1
 NO_RAZAO_SOCIAL              | JOSE R VALENTE JR CLINICA ODONTOLOGICA ME
 NO_FANTASIA                  | ESTETICA ORAL VALENTE
 NO_LOGRADOURO                | RUA SERRA DE BOTUCATU
 NU_ENDERECO                  | 878
 NO_COMPLEMENTO               | SALAS 1102 1103 1104
 NO_BAIRRO                    | VILA GOMES CARDIM
 CO_CEP                       | 03317000
 CO_REGIAO_SAUDE              | NULL
 CO_MICRO_REGIAO              | NULL
 CO_DISTRITO_SANITARIO        | NULL
 CO_DISTRITO_ADMINISTRATIVO   | NULL
 NU_TELEFONE                  | 3582-4952
 NU_FAX                       | NULL
 NO_EMAIL                     | fiscal_reobote@hotmail.com
 NU_CPF                       | NULL
 NU_CNPJ                      | 23776096000180
 CO_ATIVIDADE                 | 04
 CO_CLIENTELA                 | 01
 NU_ALVARA                    | 339716
 DT_EXPEDICAO                 | 17-fev-2016 00:00:00
 TP_ORGAO_EXPEDIDOR           | 2
 DT_VAL_LIC_SANI              | NULL
 TP_LIC_SANI                  | NULL
 CO_TURNO_ATENDIMENTO         | 03
 CO_ESTADO_GESTOR             | 35
 CO_MUNICIPIO_GESTOR          | 355030
 DT_ATUALIZACAO               | 25/02/2016
 CO_USUARIO                   | CADASTRO
 CO_CPFDIRETORCLN             | 30711122890
 REG_DIRETORCLN               | 84240
 ST_ADESAO_FILANTROP          | NULL
 CO_MOTIVO_DESAB              | 08
 NO_URL                       | NULL
 NU_LATITUDE                  | NULL
 NU_LONGITUDE                 | NULL
 DT_ATU_GEO                   | NULL
 NO_USUARIO_GEO               | NULL
 CO_NATUREZA_JUR              | 2305
```

```
 TP_ESTAB_SEMPRE_ABERTO       | N
 ST_GERACREDITO_GERENTE_SGIF  | NULL
 ST_CONEXAO_INTERNET          | S
 CO_TIPO_UNIDADE              | NULL
 NO_FANTASIA_ABREV            | NULL
 TP_GESTAO                    | M
 DT_ATUALIZACAO_ORIGEM        | 11/03/2016
 CO_ATIVIDADE_PRINCIPAL       | NULL
 ST_CONTRATO_FORMALIZADO      | NULL
 TP_UNIDADE_DESC              | CLINICA/CENTRO DE ESPECIALIDADE
 CO_TIPO_ESTABELECIMENTO_DESC | NULL
only showing top 1 row
```

[34]:
```
# Salva a tabela `cnes_df` na pasta do data warehouse `sus-data-warehouse` em␣
 ↪formato Apache Parquet

cnes_df.write \
    .format('parquet') \
    .mode('overwrite') \
    .save('sus-data-warehouse/cnes')
```

1.3.3 Dimensão SIGTAP - PROCEDIMENTOS (sigtap_proced_df) (sia_df['PA_PROC_ID'])

[35]:
```
# Carrega num dataframe os arquivos da tabela de dimensão SIGTAP - PROCEDIMENTOS

sigtap_proced_df = spark.read.option('mergeSchema', 'true').parquet('data/
 ↪tb_sigtap_procedimento')
```

[36]:
```
# Exibe o primeiro registro da tabela SIGTAP - PROCEDIMENTOS
sigtap_proced_df.show(1, vertical=True, truncate=False)
```

```
-RECORD
0-------------------------------------------------------------------------
 CO_PROCEDIMENTO      | 0101010010
 NO_PROCEDIMENTO      | ATIVIDADE EDUCATIVA / ORIENTAÇÃO EM GRUPO NA ATENÇÃO
PRIMÁRIA
 TP_COMPLEXIDADE      | 1
 TP_SEXO              | N
 QT_MAXIMA_EXECUCAO   | 9999
 QT_DIAS_PERMANENCIA  | 9999
 QT_PONTOS            | 0000
 VL_IDADE_MINIMA      | 9999
 VL_IDADE_MAXIMA      | 9999
 VL_SH                | 0000000000
 VL_SA                | 0000000000
 VL_SP                | 0000000000
```

```
 CO_FINANCIAMENTO     | 01
 CO_RUBRICA           | NULL
 QT_TEMPO_PERMANENCIA | 9999
 DT_COMPETENCIA       | 202009
only showing top 1 row
```

[37]:
```
# Renomeia a coluna 'CO_PROCEDIMENTO' para 'PA_PROC_ID' para padronizar com a␣
↪tabela de fatos SIA_PA

sigtap_proced_df = sigtap_proced_df \
    .withColumnRenamed('CO_PROCEDIMENTO', 'PA_PROC_ID')
```

[38]:
```
# Salva a tabela `sigtap_proced_df` na pasta do data warehouse␣
↪`sus-data-warehouse` em formato Apache Parquet

sigtap_proced_df.write \
    .format('parquet') \
    .mode('overwrite') \
    .save('sus-data-warehouse/sigtap_proced')
```

1.3.4 Dimensão CID - CÓDIGO INTERNACIONAL DE DOENÇAS (cid_df) (sia_df['PA_CIDPRI', 'PA_CIDSEC', 'PA_CIDCAS'])

[39]:
```
# Carrega num dataframe a tabela CID-10 - Código Internacional de Doenças

cid_df = spark.read.option('mergeSchema', 'true').parquet('data/tb_sigtap_cid')
```

[40]:
```
# Conta a quantidade registros

cid_df.count()
```

[40]: 14230

[41]:
```
# Conta o número de códigos CID únicos

cid_df \
    .select('CO_CID') \
    .distinct() \
    .count()
```

[41]: 14230

[42]:
```
# Mostra as colunas e tipos de dados no dataframe CID

cid_df.printSchema()
```

```
root
 |-- CO_CID: string (nullable = true)
 |-- NO_CID: string (nullable = true)
 |-- TP_AGRAVO: string (nullable = true)
 |-- TP_SEXO: string (nullable = true)
 |-- TP_ESTADIO: string (nullable = true)
 |-- VL_CAMPOS_IRRADIADOS: string (nullable = true)
```

[43]:
```
# Mostra os primeiros 20 registros do dataframe CID

cid_df.show()
```

```
+------+--------------------+---------+-------+----------+--------------------+
|CO_CID|              NO_CID|TP_AGRAVO|TP_SEXO|TP_ESTADIO|VL_CAMPOS_IRRADIADOS|
+------+--------------------+---------+-------+----------+--------------------+
|   A00|              Cólera|        0|      I|         N|                 000|
|  A000|Cólera devida a V…|        2|      I|         N|                 000|
|  A001|Cólera devida a V…|        2|      I|         N|                 000|
|  A009|Cólera não especi…|        2|      I|         N|                 000|
|   A01|Febres tifóide e …|        0|      I|         N|                 000|
|  A010|       Febre tifóide|        1|      I|         N|                 000|
|  A011| Febre paratifóide A|        0|      I|         N|                 000|
|  A012| Febre paratifóide B|        0|      I|         N|                 000|
|  A013| Febre paratifóide C|        0|      I|         N|                 000|
|  A014|Febre paratifóide…|        0|      I|         N|                 000|
|   A02|Outras infecções …|        0|      I|         N|                 000|
|  A020|Enterite por salm…|        1|      I|         N|                 000|
|  A021|Septicemia por sa…|        1|      I|         N|                 000|
|  A022|Infecções localiz…|        1|      I|         N|                 000|
|  A028|Outras infecções …|        1|      I|         N|                 000|
|  A029|Infecção não espe…|        1|      I|         N|                 000|
|   A03|          Shiguelose|        0|      I|         N|                 000|
|  A030|Shiguelose devida…|        0|      I|         N|                 000|
|  A031|Shiguelose devida…|        0|      I|         N|                 000|
|  A032|Shiguelose devida…|        0|      I|         N|                 000|
+------+--------------------+---------+-------+----------+--------------------+
only showing top 20 rows
```

[44]:
```
cid_df \
    .selectExpr('len(CO_CID) as LEN',
                'CO_CID',
                'NO_CID') \
    .groupBy('LEN') \
    .count() \
    .show()
```

```
+---+-----+
```

```
|LEN|count|
+---+-----+
|  3| 2042|
|  4|12188|
+---+-----+
```

[45]:
```
cid_df = cid_df \
    .withColumn('CO_CATEG', F.left('CO_CID', F.lit(3)))

cid_df.show()
```

```
+------+--------------------+---------+-------+----------+--------------------+-
-------+
|CO_CID|
NO_CID|TP_AGRAVO|TP_SEXO|TP_ESTADIO|VL_CAMPOS_IRRADIADOS|CO_CATEG|
+------+--------------------+---------+-------+----------+--------------------+-
-------+
|   A00|              Cólera|        0|      I|         N|                 000|
A00|
|  A000|Cólera devida a V…|        2|      I|         N|                 000|
A00|
|  A001|Cólera devida a V…|        2|      I|         N|                 000|
A00|
|  A009|Cólera não especi…|        2|      I|         N|                 000|
A00|
|   A01|Febres tifóide e …|        0|      I|         N|                 000|
A01|
|  A010|       Febre tifóide|        1|      I|         N|                 000|
A01|
|  A011| Febre paratifóide A|        0|      I|         N|                 000|
A01|
|  A012| Febre paratifóide B|        0|      I|         N|                 000|
A01|
|  A013| Febre paratifóide C|        0|      I|         N|                 000|
A01|
|  A014|Febre paratifóide…|        0|      I|         N|                 000|
A01|
|   A02|Outras infecções …|        0|      I|         N|                 000|
A02|
|  A020|Enterite por salm…|        1|      I|         N|                 000|
A02|
|  A021|Septicemia por sa…|        1|      I|         N|                 000|
A02|
|  A022|Infecções localiz…|        1|      I|         N|                 000|
A02|
|  A028|Outras infecções …|        1|      I|         N|                 000|
A02|
```

```
|  A029|Infecção não espe…|        1|      I|         N|                 000|
A02|
|   A03|          Shiguelose|        0|      I|         N|                 000|
A03|
|  A030|Shiguelose devida…|        0|      I|         N|                 000|
A03|
|  A031|Shiguelose devida…|        0|      I|         N|                 000|
A03|
|  A032|Shiguelose devida…|        0|      I|         N|                 000|
A03|
+------+--------------------+---------+-------+----------+--------------------+-
-------+
only showing top 20 rows
```

[46]:
```
cid_df = cid_df.alias('A') \
    .join(cid_df.select('CO_CID', 'NO_CID').alias('B'), F.col('A.CO_CATEG') ==␣
 ↪F.col('B.CO_CID')) \
    .selectExpr('A.CO_CID',
                'A.NO_CID',
                'CO_CATEG',
                'B.NO_CID as NO_CATEG',
                'TP_AGRAVO',
                'TP_SEXO',
                'TP_ESTADIO',
                'VL_CAMPOS_IRRADIADOS')

cid_df.show()
```

```
+------+--------------------+--------+--------------------+---------+-------+---
-------+--------------------+
|CO_CID|              NO_CID|CO_CATEG|
NO_CATEG|TP_AGRAVO|TP_SEXO|TP_ESTADIO|VL_CAMPOS_IRRADIADOS|
+------+--------------------+--------+--------------------+---------+-------+---
-------+--------------------+
|   A00|              Cólera|     A00|              Cólera|        0|      I|
N|                 000|
|  A000|Cólera devida a V…|     A00|              Cólera|        2|      I|
N|                 000|
|  A001|Cólera devida a V…|     A00|              Cólera|        2|      I|
N|                 000|
|  A009|Cólera não especi…|     A00|              Cólera|        2|      I|
N|                 000|
|   A01|Febres tifóide e …|     A01|Febres tifóide e …|        0|      I|
N|                 000|
|  A010|       Febre tifóide|     A01|Febres tifóide e …|        1|      I|
N|                 000|
|  A011| Febre paratifóide A|     A01|Febres tifóide e …|        0|      I|
```

```
N|                 000|
|  A012| Febre paratifóide B|     A01|Febres tifóide e …|        0|      I|
N|                 000|
|  A013| Febre paratifóide C|     A01|Febres tifóide e …|        0|      I|
N|                 000|
|  A014|Febre paratifóide…|     A01|Febres tifóide e …|        0|      I|
N|                 000|
|   A02|Outras infecções …|     A02|Outras infecções …|        0|      I|
N|                 000|
|  A020|Enterite por salm…|     A02|Outras infecções …|        1|      I|
N|                 000|
|  A021|Septicemia por sa…|     A02|Outras infecções …|        1|      I|
N|                 000|
|  A022|Infecções localiz…|     A02|Outras infecções …|        1|      I|
N|                 000|
|  A028|Outras infecções …|     A02|Outras infecções …|        1|      I|
N|                 000|
|  A029|Infecção não espe…|     A02|Outras infecções …|        1|      I|
N|                 000|
|   A03|          Shiguelose|     A03|          Shiguelose|        0|      I|
N|                 000|
|  A030|Shiguelose devida…|     A03|          Shiguelose|        0|      I|
N|                 000|
|  A031|Shiguelose devida…|     A03|          Shiguelose|        0|      I|
N|                 000|
|  A032|Shiguelose devida…|     A03|          Shiguelose|        0|      I|
N|                 000|
+------+--------------------+--------+--------------------+---------+-------+---
-------+--------------------+
only showing top 20 rows
```

[47]:
```
# Salva a tabela `cid_df` na pasta do data warehouse `sus-data-warehouse` em␣
↪formato Apache Parquet

cid_df.write \
    .format('parquet') \
    .mode('overwrite') \
    .save('sus-data-warehouse/cid')
```

1.3.5 Dimensão ANO_MES (ano_mes_df) (sia_df['PA_CMP'])

[48]:
```
# Cria um data frame auxiliar para a (sub)dimensão MÊS

mes_df = spark.createDataFrame(
    data=[('01', 'JANEIRO', 'JAN', '1', '1'),
          ('02', 'FEVEREIRO', 'FEV', '1', '1'),
```

```
          ('03', 'MARÇO', 'MAR', '1', '1'),
          ('04', 'ABRIL', 'ABR', '2', '1'),
          ('05', 'MAIO', 'MAI', '2', '1'),
          ('06', 'JUNHO', 'JUN', '2', '1'),
          ('07', 'JULHO', 'JUL', '3', '2'),
          ('08', 'AGOSTO', 'AGO', '3', '2'),
          ('09', 'SETEMBRO', 'SET', '3', '2'),
          ('10', 'OUTUBRO', 'OUT', '4', '2'),
          ('11', 'NOVEMBRO', 'NOV', '4', '2'),
          ('12', 'DEZEMBRO', 'DEZ', '4', '2')],
    schema=['MES', 'MES_NOME', 'MES_ABREV', 'TRIMESTRE', 'SEMESTRE']
)

mes_df.show()
```

```
+---+---------+---------+---------+--------+
|MES| MES_NOME|MES_ABREV|TRIMESTRE|SEMESTRE|
+---+---------+---------+---------+--------+
| 01|  JANEIRO|      JAN|        1|       1|
| 02|FEVEREIRO|      FEV|        1|       1|
| 03|    MARÇO|      MAR|        1|       1|
| 04|    ABRIL|      ABR|        2|       1|
| 05|     MAIO|      MAI|        2|       1|
| 06|    JUNHO|      JUN|        2|       1|
| 07|    JULHO|      JUL|        3|       2|
| 08|   AGOSTO|      AGO|        3|       2|
| 09| SETEMBRO|      SET|        3|       2|
| 10|  OUTUBRO|      OUT|        4|       2|
| 11| NOVEMBRO|      NOV|        4|       2|
| 12| DEZEMBRO|      DEZ|        4|       2|
+---+---------+---------+---------+--------+
```

[49]:
```
# Cria uma lista de string de anos e meses no formato AAAAMM de 2015 a 2020

pa_cmp = [str(ano) + (str(mes) if len(str(mes)) > 1 else '0' + str(mes))
          for ano in list(range(2015, 2021))
          for mes in list(range(1, 13))]
```

[50]:
```
# Cria o data frame para a dimensão ANO_MES e popula com a informação dos meses

ano_mes_df = spark.createDataFrame(
    data=[(ano_mes,) for ano_mes in pa_cmp],
    schema=['PA_CMP']
)

ano_mes_df = ano_mes_df \
    .selectExpr('PA_CMP',
```

```
                'left(PA_CMP, 4) as ANO',
                'right(PA_CMP, 2) as MES') \
    .join(mes_df, 'MES') \
    .orderBy('PA_CMP') \
    .select('PA_CMP', 'ANO', 'MES', 'MES_NOME', 'MES_ABREV', 'TRIMESTRE',␣
 ↪'SEMESTRE')

ano_mes_df.show()
```

```
+------+----+---+---------+---------+---------+--------+
|PA_CMP| ANO|MES| MES_NOME|MES_ABREV|TRIMESTRE|SEMESTRE|
+------+----+---+---------+---------+---------+--------+
|201501|2015| 01|  JANEIRO|      JAN|        1|       1|
|201502|2015| 02|FEVEREIRO|      FEV|        1|       1|
|201503|2015| 03|    MARÇO|      MAR|        1|       1|
|201504|2015| 04|    ABRIL|      ABR|        2|       1|
|201505|2015| 05|     MAIO|      MAI|        2|       1|
|201506|2015| 06|    JUNHO|      JUN|        2|       1|
|201507|2015| 07|    JULHO|      JUL|        3|       2|
|201508|2015| 08|   AGOSTO|      AGO|        3|       2|
|201509|2015| 09| SETEMBRO|      SET|        3|       2|
|201510|2015| 10|  OUTUBRO|      OUT|        4|       2|
|201511|2015| 11| NOVEMBRO|      NOV|        4|       2|
|201512|2015| 12| DEZEMBRO|      DEZ|        4|       2|
|201601|2016| 01|  JANEIRO|      JAN|        1|       1|
|201602|2016| 02|FEVEREIRO|      FEV|        1|       1|
|201603|2016| 03|    MARÇO|      MAR|        1|       1|
|201604|2016| 04|    ABRIL|      ABR|        2|       1|
|201605|2016| 05|     MAIO|      MAI|        2|       1|
|201606|2016| 06|    JUNHO|      JUN|        2|       1|
|201607|2016| 07|    JULHO|      JUL|        3|       2|
|201608|2016| 08|   AGOSTO|      AGO|        3|       2|
+------+----+---+---------+---------+---------+--------+
only showing top 20 rows
```

[51]:
```
# Salva a tabela `ano_mes_df` na pasta do data warehouse `sus-data-warehouse`␣
 ↪em formato Apache Parquet

ano_mes_df.write \
    .format('parquet') \
    .mode('overwrite') \
    .save('sus-data-warehouse/ano_mes')
```

1.3.6 Dimensão CÓDIGO BRASILEIRO DE OCUPAÇÕES (cbocod_df) (sia_df['PA_CBOCOD'])

[52]:
```
# Instala e import o pacote `dbfread`, para carregar arquivos tipo .dbf
# Necessário para adquirir a lista do Código Brasileiro de Ocupações␣
↪disponibilizada pelo DATASUS

!pip install dbfread
from dbfread import DBF
```

```
Requirement already satisfied: dbfread in c:\mba\tcc\venv\lib\site-packages
(2.0.7)
```

[53]:
```
# Carrega a tabela a do Código Brasileiro de Ocupações (PA_CBOCOD) extraídos do␣
↪arquivo de conversão para Tabwin CBO.DBF
# baixado na página https://datasus.saude.gov.br/transferencia-de-arquivos/ em␣
↪12/08/2024
# Fonte: SIASUS - Sistema de Informações Ambulatoriais dos SUS), Modalidade:␣
↪Arquivos auxiliares para tabulação, Tipo de Arquivo: Arquivo de definição do␣
↪Tabwin
# /TAB_SIA/DBF/CBO.DBF

cbo = DBF('data/CBO.dbf', load=True)

cbo.records[0]
```

[53]:
```
OrderedDict([('CBO', '010105'), ('DS_CBO', 'Oficial General da Aeronautica')])
```

[54]:
```
# Cria o Data Frame da dimensão PA_CBOCOD - Código Brasileiro de Ocupações

cbocod_df = spark.createDataFrame(
    data=[(record['CBO'], record['DS_CBO']) for record in cbo.records],
    schema=['PA_CBOCOD', 'CBOCOD_DESC'])

cbocod_df.show(truncate=False)
```

```
+---------+-----------------------------------+
|PA_CBOCOD|CBOCOD_DESC                        |
+---------+-----------------------------------+
|010105   |Oficial General da Aeronautica     |
|010110   |Oficial General do Exercito        |
|010115   |Oficial General da Marinha         |
|010205   |Oficial da Aeronautica             |
|010210   |Oficial do Exercito                |
|010215   |Oficial da Marinha                 |
|010305   |Praca da Aeronautica               |
|010310   |Praca do Exercito                  |
|010315   |Praca da Marinha                   |
```

```
|020105   |Coronel da Policia Militar         |
|020110   |Tenente-Coronel da Policia Militar |
|020115   |Major da Policia Militar           |
|020205   |Capitao da Policia Militar         |
|020305   |Primeiro Tenente de Policia Militar|
|020310   |Segundo Tenente de Policia Militar |
|021105   |Subtenente da Policia Militar      |
|021110   |Sargento da Policia Militar        |
|021205   |Cabo da Policia Militar            |
|021210   |Soldado da Policia Militar         |
|030105   |Coronel Bombeiro Militar           |
+---------+-----------------------------------+
only showing top 20 rows
```

[55]:
```
# Quantidade de registros no data frame 'cbocod_df'

cbocod_df.count()
```

[55]: 2812

[56]:
```
# Salva a tabela `ano_mes_df` na pasta do data warehouse `sus-data-warehouse`␣
 ↪em formato Apache Parquet

cbocod_df.write \
    .format('parquet') \
    .mode('overwrite') \
    .save('sus-data-warehouse/cbocod')
```

1.3.7 Dimensão TIPO DE ESTABELECIMENTO (`tpups_df`) (`sia_df['PA_TPUPS']`)

[57]:
```
# Cria um Data Frame para a dimensão Tipos de Estabelecimentos (PA_TPUPS)
# SIA.PA_TPUPS == CNES.TP_UNIDADE

# Tipos de estabelecimento (PA_TPUPS) extraídos do arquivo de conversão para␣
 ↪Tabwin TP_ESTAB.CNV
# baixado na página https://datasus.saude.gov.br/transferencia-de-arquivos/ em␣
 ↪12/08/2024
# Fonte: SIASUS - Sistema de Informações Ambulatoriais dos SUS), Modalidade:␣
 ↪Arquivos auxiliares para tabulação, Tipo de Arquivo: Arquivo de definição do␣
 ↪Tabwin
# /TAB_SIA/CNV/TP_ESTAB.CNV

tpups_df = spark.createDataFrame(
    data=[('74', 'ACADEMIA DA SAÚDE'),
          ('81', 'CENTRAL DE REGUALAÇÃO'),
          ('76', 'CENTRAL DE REGULAÇÃO MÉDICA DAS URGÊNCIAS'),
```

```
            ('71', 'CENTRO DE APOIO A SAÚDE DA FAMÍLIA-CASF'),
            ('69', 'CENTRO DE ATENÇÃOO HEMOTERÁPICA E/OU HEMATOLÓGICA'),
            ('70', 'CENTRO DE ATENÇÃO PSICOSSOCIAL-CAPS'),
            ('61', 'CENTRO DE PARTO NORMAL'),
            ('02', 'CENTRO DE SAUDE/UNIDADE BASICA DE SAUDE'),
            ('64', 'CENTRAL DE REGULACAO DE SERVICOS DE SAUDE'),
            ('36', 'CLINICA ESPECIALIZADA/AMBULATORIO ESPECIALIZADO'),
            ('22', 'CONSULTORIO'),
            ('60', 'COOPERATIVA'),
            ('43', 'FARMACIA'),
            ('07', 'HOSPITAL ESPECIALIZADO'),
            ('05', 'HOSPITAL GERAL'),
            ('62', 'HOSPITAL DIA'),
            ('67', 'LABORATORIO CENTRAL DE SAUDE PUBLICA - LACEN'),
            ('80', 'ORIO DE SAUDE PUBLICA'),
            ('04', 'POLICLINICA'),
            ('79', 'OFICINA ORTOPEDICA'),
            ('01', 'POSTO DE SAUDE'),
            ('73', 'PRONTO ANTEDIMENTO'),
            ('21', 'PRONTO SOCORRO ESPECIALIZADO'),
            ('20', 'PRONTO SOCORRO GERAL'),
            ('68', 'SECRETARIA DE SAUDE'),
            ('77', 'SERVICO DE ATENCAO DOMICILIAR ISOLADO(HOME CARE)'),
            ('63', 'UNIDADE AUTORIZADORA'),
            ('72', 'UNIDADE DE ATENÇÃO E SAÚDE INDÍGENA'),
            ('78', 'UNIDADE DE ATENCAO EM REGIME RESIDENCIAL'),
            ('39', 'UNIDADE DE SERVICO DE APOIO DE DIAGNOSE E TERAPIA'),
            ('45', 'UNIDADE DE SAUDE DA FAMILIA'),
            ('50', 'UNIDADE DE VIGILANCIA EM SAUDE'),
            ('65', 'UNIDADE DE VIGILANCIA EPIDEMIOLOGIA (ANTIGO)'),
            ('66', 'UNIDADE DE VIGILANCIA SANITARIA (ANTIGO)'),
            ('15', 'UNIDADE MISTA'),
            ('42', 'UNIDADE MOVEL DE NIVEL PRE-HOSP-URGENCIA/EMERGENCIA'),
            ('32', 'UNIDADE MOVEL FLUVIAL'),
            ('40', 'UNIDADE MOVEL TERRESTRE'),
            ('75', 'TELESAÚDE'),
            ('09', 'PRONTO SOCORRO DE HOSPITAL GERAL (ANTIGO)'),
            ('12', 'PRONTO SOCORRO TRAUMATO-ORTOPEDICO (ANTIGO)'),
            ('-99', 'TIPO ESTABELECIMENTO NÃO INFORMADO')],
    schema=['PA_TPUPS', 'TPUPS_DESC']
)

tpups_df.show(truncate=False)
```

```
+--------+-------------------------------------------------+
|PA_TPUPS|TPUPS_DESC                                       |
+--------+-------------------------------------------------+
|74      |ACADEMIA DA SAÚDE                                |
```

```
|81      |CENTRAL DE REGUALAÇÃO                                |
|76      |CENTRAL DE REGULAÇÃO MÉDICA DAS URGÊNCIAS            |
|71      |CENTRO DE APOIO A SAÚDE DA FAMÍLIA-CASF              |
|69      |CENTRO DE ATENÇÃOO HEMOTERÁPICA E/OU HEMATOLÓGICA    |
|70      |CENTRO DE ATENÇÃO PSICOSSOCIAL-CAPS                  |
|61      |CENTRO DE PARTO NORMAL                               |
|02      |CENTRO DE SAUDE/UNIDADE BASICA DE SAUDE              |
|64      |CENTRAL DE REGULACAO DE SERVICOS DE SAUDE            |
|36      |CLINICA ESPECIALIZADA/AMBULATORIO ESPECIALIZADO      |
|22      |CONSULTORIO                                          |
|60      |COOPERATIVA                                          |
|43      |FARMACIA                                             |
|07      |HOSPITAL ESPECIALIZADO                               |
|05      |HOSPITAL GERAL                                       |
|62      |HOSPITAL DIA                                         |
|67      |LABORATORIO CENTRAL DE SAUDE PUBLICA - LACEN         |
|80      |ORIO DE SAUDE PUBLICA                                |
|04      |POLICLINICA                                          |
|79      |OFICINA ORTOPEDICA                                   |
+--------+-----------------------------------------------------+
only showing top 20 rows
```

[58]:
```
# Salva a tabela `tpups_df` na pasta do data warehouse `sus-data-warehouse` em␣
↪formato Apache Parquet

tpups_df.write \
    .format('parquet') \
    .mode('overwrite') \
    .save('sus-data-warehouse/tpups')
```

1.3.8 Dimensão CARÁTER DE ATENDIMENTO (`catend_df`) (`sia_df['PA_CATEND']`)

[59]:
```
# Caráteres de Atendimento (PA_CATEND) extraídos do arquivo de conversão para␣
↪Tabwin CARAT_AT.CNV
# baixado na página https://datasus.saude.gov.br/transferencia-de-arquivos/ em␣
↪12/08/2024
# Fonte: SIASUS - Sistema de Informações Ambulatoriais dos SUS), Modalidade:␣
↪Arquivos auxiliares para tabulação, Tipo de Arquivo: Arquivo de definição do␣
↪Tabwin
# /TAB_SIA/CNV/CARAT_AT.CNV

catend_df = spark.createDataFrame(
    data=[
        ('01', 'ELETIVO'),
        ('02', 'URGÊNCIA'),
        ('03', 'ACIDENTE NO LOCAL TRABALHO OU A SERViÇO DA EMPRESA'),
```

```
        ('04', 'ACIDENTE NO TRAJETO PARA O TRABALHO '),
        ('05', 'OUTROS TIPOS DE ACIDENTE DE TRÂNSITO'),
        ('06', 'OUTROS TIPOS LESÕES/ENVENENAMENTOS(AGENT.FIS./QUIM.'),
        ('99', 'INFORMAÇÃO INEXISTENTE  (BPA-C)'),
        ('00', 'CARATER DE ATENDIMENTO NÃO INFORMADO'),
        ('07', 'CARATER DE ATENDIMENTO INVALIDO'),
        ('10', 'CARATER DE ATENDIMENTO INVALIDO'),
        ('12', 'CARATER DE ATENDIMENTO INVALIDO'),
        ('20', 'CARATER DE ATENDIMENTO INVALIDO'),
        ('53', 'CARATER DE ATENDIMENTO INVALIDO'),
        ('54', 'CARATER DE ATENDIMENTO INVALIDO'),
        ('57', 'CARATER DE ATENDIMENTO INVALIDO'),
        ('0-', 'CARATER DE ATENDIMENTO INVALIDO'),
        ('0E', 'CARATER DE ATENDIMENTO INVALIDO'),
        ('0U', 'CARATER DE ATENDIMENTO INVALIDO')
    ],
    schema=['PA_CATEND', 'CATEND_DESC']
)

catend_df.show(truncate=False)
```

```
+---------+-------------------------------------------------+
|PA_CATEND|CATEND_DESC                                      |
+---------+-------------------------------------------------+
|01       |ELETIVO                                          |
|02       |URGÊNCIA                                         |
|03       |ACIDENTE NO LOCAL TRABALHO OU A SERViÇO DA EMPRESA |
|04       |ACIDENTE NO TRAJETO PARA O TRABALHO              |
|05       |OUTROS TIPOS DE ACIDENTE DE TRÂNSITO             |
|06       |OUTROS TIPOS LESÕES/ENVENENAMENTOS(AGENT.FIS./QUIM.|
|99       |INFORMAÇÃO INEXISTENTE  (BPA-C)                  |
|00       |CARATER DE ATENDIMENTO NÃO INFORMADO             |
|07       |CARATER DE ATENDIMENTO INVALIDO                  |
|10       |CARATER DE ATENDIMENTO INVALIDO                  |
|12       |CARATER DE ATENDIMENTO INVALIDO                  |
|20       |CARATER DE ATENDIMENTO INVALIDO                  |
|53       |CARATER DE ATENDIMENTO INVALIDO                  |
|54       |CARATER DE ATENDIMENTO INVALIDO                  |
|57       |CARATER DE ATENDIMENTO INVALIDO                  |
|0-       |CARATER DE ATENDIMENTO INVALIDO                  |
|0E       |CARATER DE ATENDIMENTO INVALIDO                  |
|0U       |CARATER DE ATENDIMENTO INVALIDO                  |
+---------+-------------------------------------------------+
```

[60]:
```
# Salva a tabela `catend_df` na pasta do data warehouse `sus-data-warehouse` em␣
↪formato Apache Parquet
```

```
catend_df.write \
    .format('parquet') \
    .mode('overwrite') \
    .save('sus-data-warehouse/catend')
```

1.3.9 Dimensão DOCUMENTO DE ORIGEM (docorig_df) (sia_df['PA_DOCORIG'])

[61]:
```
# Documentos de Origem (PA_DOCORIG) extraídos do arquivo de conversão para Tabwin DOCORIG.CNV
# baixado na página https://datasus.saude.gov.br/transferencia-de-arquivos/ em 12/08/2024
# Fonte: SIASUS - Sistema de Informações Ambulatoriais dos SUS), Modalidade: Arquivos auxiliares para tabulação, Tipo de Arquivo: Arquivo de definição do Tabwin
# /TAB_SIA/CNV/DOCORIG.CNV

docorig_df = spark.createDataFrame(
    data=[
        ('C', 'BPA - Consolidado'),  # BPA-C
        ('I', 'BPA - Individualizado'),  # BPA-I
        ('P', 'APAC - Procedimento Principal'),
        ('S', 'APAC - Procedimento Secundário'),
        ('A', 'RAAS - Atenção Domiciliar'),
        ('B', 'RAAS - Psicossocial')
    ],
    schema=['PA_DOCORIG', 'DOCORIG_DESC']
)

docorig_df.show(truncate=False)
```

```
+----------+------------------------------+
|PA_DOCORIG|DOCORIG_DESC                  |
+----------+------------------------------+
|C         |BPA - Consolidado             |
|I         |BPA - Individualizado         |
|P         |APAC - Procedimento Principal |
|S         |APAC - Procedimento Secundário|
|A         |RAAS - Atenção Domiciliar     |
|B         |RAAS - Psicossocial           |
+----------+------------------------------+
```

[62]:
```
# Quantidade total de registros do data frame 'sia_df'

sia_df_count = sia_df.count()
```

[63]:
```
# Verifica a distribuição de registros por PA_DOCORIG

sia_df \
    .groupBy('PA_DOCORIG') \
    .count() \
    .withColumn('DOCORIG_PERCENT', F.round(F.col('count') / sia_df_count * 100,␣
 ↪2)) \
    .join(docorig_df, 'PA_DOCORIG') \
    .orderBy(F.desc('count')) \
    .withColumn('count', F.format_number('count', 0)) \
    .show(truncate=False)
```

```
+----------+-----------+---------------+------------------------------+
|PA_DOCORIG|count      |DOCORIG_PERCENT|DOCORIG_DESC                  |
+----------+-----------+---------------+------------------------------+
|I         |958,547,600|55.0           |BPA - Individualizado         |
|C         |460,061,443|26.4           |BPA - Consolidado             |
|P         |147,446,676|8.46           |APAC - Procedimento Principal |
|S         |119,552,212|6.86           |APAC - Procedimento Secundário|
|B         |56,971,394 |3.27           |RAAS - Psicossocial           |
|A         |164,644    |0.01           |RAAS - Atenção Domiciliar     |
+----------+-----------+---------------+------------------------------+
```

[64]:
```
sia_df \
    .where('PA_DOCORIG = "C"') \
    .show(1, vertical=True)
```

```
-RECORD 0--------------------
 PA_CMP     | 201907
 PA_CODUNI  | 4026896
 PA_TPUPS   | 05
 PA_UFMUN   | 291460
 PA_PROC_ID | 0202010635
 PA_DOCORIG | C
 PA_CNSMED  | 000000000000000
 PA_CBOCOD  | 223415
 PA_MOTSAI  | 00
 PA_CIDPRI  | 0000
 PA_CIDSEC  | 0000
 PA_CIDCAS  | 0000
 PA_CATEND  | 99
 PA_IDADE   | 999
 PA_SEXO    | 0
 PA_RACACOR | 00
 PA_MUNPCN  | 999999
 PA_QTDAPR  | 1
 PA_VALAPR  | 1.85
```

```
only showing top 1 row
```

[65]:
```
# Salva a tabela `docorig_df` na pasta do data warehouse `sus-data-warehouse`
↪em formato Apache Parquet

docorig_df.write \
    .format('parquet') \
    .mode('overwrite') \
    .save('sus-data-warehouse/docorig')
```

1.3.10 Dimensão SEXO (sexo_df) (sia_df['PA_SEXO'])

[66]:
```
# Verifica a contagem de valores na coluna PA_SEXO

sia_df \
    .groupBy('PA_SEXO') \
    .count() \
    .orderBy(F.desc('count')) \
    .withColumn('count', F.format_number('count', 0)) \
    .show()
```

```
+-------+-----------+
|PA_SEXO|      count|
+-------+-----------+
|      F|761,155,673|
|      M|521,526,726|
|      0|460,061,550|
|     01|          8|
|     03|          7|
|     99|          5|
+-------+-----------+
```

PA_SEXO “0”? Não informado? [‘01’, ‘02’, ‘99’] devem ser erros.

[67]:
```
# Sexos (PA_SEXO) extraídos do arquivo de conversão para Tabwin SEXO.CNV
# baixado na página https://datasus.saude.gov.br/transferencia-de-arquivos/ em
↪12/08/2024
# Fonte: SIASUS - Sistema de Informações Ambulatoriais dos SUS), Modalidade:
↪Arquivos auxiliares para tabulação, Tipo de Arquivo: Arquivo de definição do
↪Tabwin
# /TAB_SIA/CNV/SEXO.CNV

sexo_df = spark.createDataFrame(
    data= [('0', 'Não exigido'),
           ('M', 'Masculino'),
           ('F', 'Feminino')],
```

```
    schema=['PA_SEXO', 'SEXO_DESC']
)

sexo_df.show()
```

```
+-------+-----------+
|PA_SEXO|  SEXO_DESC|
+-------+-----------+
|      0|Não exigido|
|      M|  Masculino|
|      F|   Feminino|
+-------+-----------+
```

[68]:
```
# Salva a tabela `sexo_df` na pasta do data warehouse `sus-data-warehouse` em␣
↪formato Apache Parquet

sexo_df.write \
    .format('parquet') \
    .mode('overwrite') \
    .save('sus-data-warehouse/sexo')
```

1.3.11 Dimensão RAÇA/COR (raca_cor_df) (sia_df['PA_RACACOR'])

[69]:
```
# Raça/Cor (PA_RACACOR) extraídos do arquivo de conversão para Tabwin RACA_COR.
↪CNV
# baixado na página https://datasus.saude.gov.br/transferencia-de-arquivos/ em␣
↪12/08/2024
# Fonte: SIASUS - Sistema de Informações Ambulatoriais dos SUS), Modalidade:␣
↪Arquivos auxiliares para tabulação, Tipo de Arquivo: Arquivo de definição do␣
↪Tabwin
# /TAB_SIA/CNV/RACA_COR.CNV

raca_cor_df = spark.createDataFrame(
    data=[('01', 'BRANCA'),
          ('02', 'PRETA'),
          ('03', 'PARDA'),
          ('04', 'AMARELA'),
          ('05', 'INDIGENA'),
          ('99', 'SEM INFORMAÇÃO'),
          ('06', 'RAÇA/COR=06 (INDEVIDO)'),
          ('09', 'RAÇA/COR=09 (INDEVIDO)'),
          ('1M', 'RACA/COR (OUTROS INDEVIDOS)'),
          ('1G', 'RACA/COR (OUTROS INDEVIDOS)'),
          ('1C', 'RACA/COR (OUTROS INDEVIDOS)'),
          ('DE', 'RACA/COR (OUTROS INDEVIDOS)'),
          ('D', 'RACA/COR (OUTROS INDEVIDOS)'),
```

```
          ('87', 'RACA/COR (OUTROS INDEVIDOS)')],
    schema=['PA_RACACOR', 'RACACOR_DESC']
)

raca_cor_df.show(truncate=False)
```

```
+----------+--------------------------+
|PA_RACACOR|RACACOR_DESC              |
+----------+--------------------------+
|01        |BRANCA                    |
|02        |PRETA                     |
|03        |PARDA                     |
|04        |AMARELA                   |
|05        |INDIGENA                  |
|99        |SEM INFORMAÇÃO            |
|06        |RAÇA/COR=06 (INDEVIDO)    |
|09        |RAÇA/COR=09 (INDEVIDO)    |
|1M        |RACA/COR (OUTROS INDEVIDOS)|
|1G        |RACA/COR (OUTROS INDEVIDOS)|
|1C        |RACA/COR (OUTROS INDEVIDOS)|
|DE        |RACA/COR (OUTROS INDEVIDOS)|
|D         |RACA/COR (OUTROS INDEVIDOS)|
|87        |RACA/COR (OUTROS INDEVIDOS)|
+----------+--------------------------+
```

[70]:
```
# Salva a tabela `raca_cor_df` na pasta do data warehouse `sus-data-warehouse`
↪em formato Apache Parquet

raca_cor_df.write \
    .format('parquet') \
    .mode('overwrite') \
    .save('sus-data-warehouse/raca_cor')
```

1.3.12 Dimensão MOTIVO DE SAÍDA (`motsai_df`) (`sia_df['PA_MOTSAI']`)

[71]:
```
# Verifica a contagem de valores na coluna PA_MOTSAI

sia_df \
    .groupBy('PA_MOTSAI') \
    .count() \
    .orderBy(F.desc('count')) \
    .withColumn('count', F.format_number('count', 0)) \
    .show()
```

```
+---------+-------------+
|PA_MOTSAI|        count|
+---------+-------------+
```

```
|       00|1,418,609,023|
|       21|  275,660,569|
|       28|   21,568,731|
|       15|    8,617,676|
|       12|    6,889,036|
|       18|    4,178,059|
|       11|    2,269,761|
|       51|    1,272,749|
|       41|    1,063,127|
|       26|    1,006,297|
|       31|      885,177|
|       25|      178,617|
|       16|      165,496|
|       43|       98,692|
|       22|       97,093|
|       24|       70,733|
|       42|       68,800|
|       14|       34,744|
|       23|        6,243|
|       27|        3,326|
+---------+-------------+
only showing top 20 rows
```

[72]:
```
# Motivos de Saída (PA_MOTSAI) extraídos do arquivo de conversão para Tabwin␣
 ↪MOTSAIPE.CNV
# baixado na página https://datasus.saude.gov.br/transferencia-de-arquivos/ em␣
 ↪12/08/2024
# Fonte: SIASUS - Sistema de Informações Ambulatoriais dos SUS), Modalidade:␣
 ↪Arquivos auxiliares para tabulação, Tipo de Arquivo: Arquivo de definição do␣
 ↪Tabwin
# /TAB_SIA/CNV/MOTSAIPE.CNV

motsai_df = spark.createDataFrame(
    data=[('11', 'ALTA CURADO'),
          ('12', 'ALTA MELHORADO'),
          ('13', 'ALTA DA PUÉRPERA E PERMANÊNCIA DO RECÉM NASCIDO'),
          ('14', 'ALTA A PEDIDO'),
          ('15', 'ALTA COM PREVISÃO DE RETORNO P/ ACOMPAN. DO PACIENT'),
          ('16', 'ALTA POR EVASÃO'),
          ('17', 'ALTA DA PUÉRPERA E RECÉM NASCIDO'),
          ('18', 'ALTA POR OUTROS MOTIVOS'),
          ('21', 'PERMANÊNCIA POR CARACTERÍSTICAS PRÓPRIAS DA DOENÇA'),
          ('22', 'PERMANÊNCIA POR INTERCORRÊNCIA'),
          ('23', 'PERMANÊNCIA POR IMPOSSIBILIDADE SÓCIO-FAMILIAR'),
          ('24', 'PERMAN. POR PROCESSO-DOAÇÃO DE ÓRGÃOS-DOADOR VIVO'),
          ('25', 'PERMAN. POR PROCESSO-DOAÇÃO DE ÓRGÃOS-DOADOR MORTO'),
```

```
          ('26', 'PERMANÊNCIA POR MUDANÇAA DE PROCEDIMENTO'),
          ('27', 'PERMANÊNCIA POR REOPERAÇÃO'),
          ('28', 'PERMANÊNCIA POR OUTROS MOTIVOS'),
          ('31', 'TRANSFERIDO PARA OUTRO ESTABELECIMENTO'),
          ('41', 'ÓBITO COM DECLARAÇÃO DE ÓBITO FORNEC. MÉDICO ASSIST'),
          ('42', 'ÓBITO COM DECLARAÇÃO DE ÓBITO FORNECIDA PELO I.M.L'),
          ('43', 'ÓBITO COM DECLARAÇÃO DE ÓBITO FORNECIDA PELO S.V.O'),
          ('51', 'ENCERRAMENTO ADMINSTRATIVO'),
          ('00', 'PRODUÇÃO SEM MOTIVO DE SAÍDA (BPA-C / BPA-I)')],
    schema=['PA_MOTSAI', 'MOTSAI_DESC']
)

motsai_df.show(25, truncate=False)
```

```
+---------+--------------------------------------------------+
|PA_MOTSAI|MOTSAI_DESC                                       |
+---------+--------------------------------------------------+
|11       |ALTA CURADO                                       |
|12       |ALTA MELHORADO                                    |
|13       |ALTA DA PUÉRPERA E PERMANÊNCIA DO RECÉM NASCIDO    |
|14       |ALTA A PEDIDO                                     |
|15       |ALTA COM PREVISÃO DE RETORNO P/ ACOMPAN. DO PACIENT|
|16       |ALTA POR EVASÃO                                   |
|17       |ALTA DA PUÉRPERA E RECÉM NASCIDO                  |
|18       |ALTA POR OUTROS MOTIVOS                           |
|21       |PERMANÊNCIA POR CARACTERÍSTICAS PRÓPRIAS DA DOENÇA |
|22       |PERMANÊNCIA POR INTERCORRÊNCIA                    |
|23       |PERMANÊNCIA POR IMPOSSIBILIDADE SÓCIO-FAMILIAR    |
|24       |PERMAN. POR PROCESSO-DOAÇÃO DE ÓRGÃOS-DOADOR VIVO |
|25       |PERMAN. POR PROCESSO-DOAÇÃO DE ÓRGÃOS-DOADOR MORTO|
|26       |PERMANÊNCIA POR MUDANÇAA DE PROCEDIMENTO          |
|27       |PERMANÊNCIA POR REOPERAÇÃO                        |
|28       |PERMANÊNCIA POR OUTROS MOTIVOS                    |
|31       |TRANSFERIDO PARA OUTRO ESTABELECIMENTO            |
|41       |ÓBITO COM DECLARAÇÃO DE ÓBITO FORNEC. MÉDICO ASSIST|
|42       |ÓBITO COM DECLARAÇÃO DE ÓBITO FORNECIDA PELO I.M.L|
|43       |ÓBITO COM DECLARAÇÃO DE ÓBITO FORNECIDA PELO S.V.O|
|51       |ENCERRAMENTO ADMINSTRATIVO                        |
|00       |PRODUÇÃO SEM MOTIVO DE SAÍDA (BPA-C / BPA-I)      |
+---------+--------------------------------------------------+
```

[73]:
```
# Salva a tabela `motsai_df` na pasta do data warehouse `sus-data-warehouse` em formato Apache Parquet

motsai_df.write \
    .format('parquet') \
```

```
    .mode('overwrite') \
    .save('sus-data-warehouse/motsai')
```

1.4 Encerramento da sessão Spark

[74]:
```
spark.stop()
```

[ ]:
```

```

APÊNDICE C – EXEMPLOS DE CONSULTAS ANALÍTICAS

O Jupyter Notebook `sus_dw_eda.ipynb` pode ser baixado em <https://github.com/DaniloGouvea/sus-dw/blob/main/sus_dw_eda.ipynb>.

sus_dw_eda

October 8, 2024

1 Criação de data warehouse para dados públicos de atendimentos ambulatoriais do SUS

Esse Jupyter notebook é parte do trabalho de conclusão de curso do MBA em Inteligência Artificial e Big Data, oferecido pelo ICMC - USP, do aluno Danilo Gouvea Silva, da Turma 3.

Parte do capítulo de Avaliação Experimental da monografia, nesse segundo notebook (`sus_dw_eda.ipynb`), após a execução do processo de ETL - "Extract, Transform and Load" - o data warehouse está organizado em 1 tabela de fatos e 12 tabelas de dimensões armazenadas em arquivos Apache Parquet.

Aqui, todas as tabelas serão carregadas e estarão prontras para a análise exploratória através de consultas analíticas. Algumas consultas analíticas serão realizadas como exemplo.

Esse notebook foi criado e utilizado localmente. Para utilizá-lo, é necessário que estejam localmente instalados o Spark, o Java e o Python. Também é recomendado a criação de um ambiente virtual Python para a instalação de todos pacotes de Python necessários, que estão contidos no arquivo de requisitos `requirements.txt`, que disponibilizado junto a esse notebook.

É importante ressaltar que o objetivo desse notebook não é demonstrar, nem guiar a instalação e configuração do Spark numa máquina local.

1.1 Criação da sessão Spark

Nessa seção, é criado uma sessão local de Spark, com apenas um nó mestre e sem nós de trabalho e gerenciador de cluster. As tarefas (tasks) serão executadas pelo driver localizado no nó mestre e utilizarão o máximo de núcleos lógicos de processamento disponíveis no processador local.

[1]:
```
# Verifica as versões instaladas de Java, Python e Spark

!java -version
!python --version
!pyspark --version
```

```
java version "1.8.0_411"
Java(TM) SE Runtime Environment (build 1.8.0_411-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.411-b09, mixed mode)

Python 3.10.5

Welcome to
      ____              __
```

```
      / __/__  ___ _____/ /__
     _\ \/ _ \/ _ `/ __/  '_/
    /___/ .__/\_,_/_/ /_/\_\   version 3.5.1
       /_/

Using Scala version 2.12.18, Java HotSpot(TM) 64-Bit Server VM, 1.8.0_381
Branch HEAD
Compiled by user heartsavior on 2024-02-15T11:24:58Z
Revision fd86f85e181fc2dc0f50a096855acf83a6cc5d9c
Url https://github.com/apache/spark
Type --help for more information.
```

[2]:
```
# Configura corretamente as variáveis de ambiente do Spark

!pip install findspark
import findspark
findspark.init()
```

```
Requirement already satisfied: findspark in c:\mba\tcc\venv\lib\site-packages
(2.0.1)
```

[3]:
```
# Cria a sessão de Spark

from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName('sus_dw_eda') \
    .master('local[*]') \
    .getOrCreate()

spark
```

[3]:
```
<pyspark.sql.session.SparkSession at 0x1afa6fd9600>
```

1.2 Carregamento das tabelas de fatos e dimensões

[4]:
```
# Importa as funções e tipos de dados do Spark

import pyspark.sql.functions as F
import pyspark.sql.types as T
```

1.2.1 Carregamento da tabela de fatos SIASUS - SERVIÇO DE INFORMAÇÕES AMBULATORIAIS DO SUS (sia_df)

[5]:
```
# Carrega a tabela de fatos SIA - Serviço de Informações Ambulatoriais

sia_df = spark.read.parquet('sus-data-warehouse/sia')
```

[6]:
```
# Conta o número total de registros na tabela de fatos

sia_df_count = sia_df.count()

print(f'Número de registros: {sia_df_count:,}')
```

```
Número de registros: 1,742,743,969
```

[7]:
```
# Exibe a lista de colunas do dataframe SIA

sia_df \
    .printSchema()
```

```
root
 |-- PA_CMP: string (nullable = true)
 |-- PA_CODUNI: string (nullable = true)
 |-- PA_TPUPS: string (nullable = true)
 |-- PA_UFMUN: string (nullable = true)
 |-- PA_PROC_ID: string (nullable = true)
 |-- PA_DOCORIG: string (nullable = true)
 |-- PA_CNSMED: string (nullable = true)
 |-- PA_CBOCOD: string (nullable = true)
 |-- PA_MOTSAI: string (nullable = true)
 |-- PA_CIDPRI: string (nullable = true)
 |-- PA_CIDSEC: string (nullable = true)
 |-- PA_CIDCAS: string (nullable = true)
 |-- PA_CATEND: string (nullable = true)
 |-- PA_IDADE: integer (nullable = true)
 |-- PA_SEXO: string (nullable = true)
 |-- PA_RACACOR: string (nullable = true)
 |-- PA_MUNPCN: string (nullable = true)
 |-- PA_QTDAPR: integer (nullable = true)
 |-- PA_VALAPR: float (nullable = true)
```

1.2.2 Carregamento das tabelas de dimensões

Nessa seção, serão carregadas as tabelas de dimensões: - `municipios_df`: listagem do IBGE de todos os munícipios brasileiros, estados, regiões e outras informações; `sia_df['PA_UFMUN]` - `cnes_df`: Cadastro Nacional de Estabelecimentos de Saúde; `sia_df['PA_CODUNI']` - `sigtap_proced_df`: listagem dos Procedimentos oferecidos pelo SUS; `sia_df['PA_PROC_ID']` - `cid_df`: CID-10 Código Internacional de Doenças; `sia_df['PA_CIDPRI', 'PA_CIDSEC', 'PA_CIDCAS']` - `ano_mes_df`: Dimensão "data"no formato AAAAMM; `sia_df['PA_CMP']` - `cbocod_df`: Código Brasileiro de Ocupações; `sia_df['PA_CBOCOD']` - `tpups_df`: Tipos de Estabelcimentos de Saúde; `sia_df['PA_TPUPS']` - `catend_df`: Caráter de Atendimento; `sia_df['PA_CATEND']` - `docorig_df`: Tipo de Documento de Origem da produção ambulatorial; `sia_df['PA_DOCORIG']` - `sexo_df`: Sexo do paciente; `sia_df['PA_SEXO']` - `raca_cor_df`: Raça/Cor do paciente; `sia_df['PA_RACACOR']` - `motsai_df`: Motivos de saída. `sia_df['PA_MOTSAI']`

[8]:
```
# Carrega as tabelas de dimensões

municipios_df = spark.read.parquet('sus-data-warehouse/municipios')
cnes_df = spark.read.parquet('sus-data-warehouse/cnes')
sigtap_proced_df = spark.read.parquet('sus-data-warehouse/sigtap_proced')
cid_df = spark.read.parquet('sus-data-warehouse/cid')
ano_mes_df = spark.read.parquet('sus-data-warehouse/ano_mes')
cbocod_df = spark.read.parquet('sus-data-warehouse/cbocod')
tpups_df = spark.read.parquet('sus-data-warehouse/tpups')
catend_df = spark.read.parquet('sus-data-warehouse/catend')
docorig_df = spark.read.parquet('sus-data-warehouse/docorig')
sexo_df = spark.read.parquet('sus-data-warehouse/sexo')
raca_cor_df = spark.read.parquet('sus-data-warehouse/raca_cor')
motsai_df = spark.read.parquet('sus-data-warehouse/motsai')
```

[9]:
```
# Verifica os 'schemas' das tabelas de dimensões

print('MUNICIPIOS:')
municipios_df.printSchema()

print('CNES:')
cnes_df.printSchema()

print('SIGTAP_PROCED:')
sigtap_proced_df.printSchema()

print('CID:')
cid_df.printSchema()

print('ANO_MES:')
ano_mes_df.printSchema()

print('CBOCOD:')
cbocod_df.printSchema()

print('TPUPS:')
tpups_df.printSchema()

print('CATEND:')
catend_df.printSchema()

print('DOCORIG:')
docorig_df.printSchema()

print('SEXO:')
sexo_df.printSchema()
```

```
print('RACA_COR:')
raca_cor_df.printSchema()

print('MOTSAI:')
motsai_df.printSchema()
```

```
MUNICIPIOS:
root
 |-- PA_UFMUN: string (nullable = true)
 |-- NOME: string (nullable = true)
 |-- MICRORREGIAO: string (nullable = true)
 |-- MESORREGIAO: string (nullable = true)
 |-- UF: string (nullable = true)
 |-- UF_NOME: string (nullable = true)
 |-- REGIAO_SIGLA: string (nullable = true)
 |-- REGIAO: string (nullable = true)
 |-- REGIAO_IMEDIATA: string (nullable = true)
 |-- REGIAO_INTERMEDIARIA: string (nullable = true)

CNES:
root
 |-- CO_TIPO_ESTABELECIMENTO: string (nullable = true)
 |-- TP_UNIDADE: string (nullable = true)
 |-- CO_UNIDADE: string (nullable = true)
 |-- PA_CODUNI: string (nullable = true)
 |-- NU_CNPJ_MANTENEDORA: string (nullable = true)
 |-- TP_PFPJ: string (nullable = true)
 |-- NIVEL_DEP: string (nullable = true)
 |-- NO_RAZAO_SOCIAL: string (nullable = true)
 |-- NO_FANTASIA: string (nullable = true)
 |-- NO_LOGRADOURO: string (nullable = true)
 |-- NU_ENDERECO: string (nullable = true)
 |-- NO_COMPLEMENTO: string (nullable = true)
 |-- NO_BAIRRO: string (nullable = true)
 |-- CO_CEP: string (nullable = true)
 |-- CO_REGIAO_SAUDE: string (nullable = true)
 |-- CO_MICRO_REGIAO: string (nullable = true)
 |-- CO_DISTRITO_SANITARIO: string (nullable = true)
 |-- CO_DISTRITO_ADMINISTRATIVO: string (nullable = true)
 |-- NU_TELEFONE: string (nullable = true)
 |-- NU_FAX: string (nullable = true)
 |-- NO_EMAIL: string (nullable = true)
 |-- NU_CPF: string (nullable = true)
 |-- NU_CNPJ: string (nullable = true)
 |-- CO_ATIVIDADE: string (nullable = true)
 |-- CO_CLIENTELA: string (nullable = true)
 |-- NU_ALVARA: string (nullable = true)
 |-- DT_EXPEDICAO: string (nullable = true)
```

```
 |-- TP_ORGAO_EXPEDIDOR: string (nullable = true)
 |-- DT_VAL_LIC_SANI: string (nullable = true)
 |-- TP_LIC_SANI: string (nullable = true)
 |-- CO_TURNO_ATENDIMENTO: string (nullable = true)
 |-- CO_ESTADO_GESTOR: string (nullable = true)
 |-- CO_MUNICIPIO_GESTOR: string (nullable = true)
 |-- DT_ATUALIZACAO: string (nullable = true)
 |-- CO_USUARIO: string (nullable = true)
 |-- CO_CPFDIRETORCLN: string (nullable = true)
 |-- REG_DIRETORCLN: string (nullable = true)
 |-- ST_ADESAO_FILANTROP: string (nullable = true)
 |-- CO_MOTIVO_DESAB: string (nullable = true)
 |-- NO_URL: string (nullable = true)
 |-- NU_LATITUDE: string (nullable = true)
 |-- NU_LONGITUDE: string (nullable = true)
 |-- DT_ATU_GEO: string (nullable = true)
 |-- NO_USUARIO_GEO: string (nullable = true)
 |-- CO_NATUREZA_JUR: string (nullable = true)
 |-- TP_ESTAB_SEMPRE_ABERTO: string (nullable = true)
 |-- ST_GERACREDITO_GERENTE_SGIF: string (nullable = true)
 |-- ST_CONEXAO_INTERNET: string (nullable = true)
 |-- CO_TIPO_UNIDADE: string (nullable = true)
 |-- NO_FANTASIA_ABREV: string (nullable = true)
 |-- TP_GESTAO: string (nullable = true)
 |-- DT_ATUALIZACAO_ORIGEM: string (nullable = true)
 |-- CO_ATIVIDADE_PRINCIPAL: string (nullable = true)
 |-- ST_CONTRATO_FORMALIZADO: string (nullable = true)
 |-- TP_UNIDADE_DESC: string (nullable = true)
 |-- CO_TIPO_ESTABELECIMENTO_DESC: string (nullable = true)

SIGTAP_PROCED:
root
 |-- PA_PROC_ID: string (nullable = true)
 |-- NO_PROCEDIMENTO: string (nullable = true)
 |-- TP_COMPLEXIDADE: string (nullable = true)
 |-- TP_SEXO: string (nullable = true)
 |-- QT_MAXIMA_EXECUCAO: string (nullable = true)
 |-- QT_DIAS_PERMANENCIA: string (nullable = true)
 |-- QT_PONTOS: string (nullable = true)
 |-- VL_IDADE_MINIMA: string (nullable = true)
 |-- VL_IDADE_MAXIMA: string (nullable = true)
 |-- VL_SH: string (nullable = true)
 |-- VL_SA: string (nullable = true)
 |-- VL_SP: string (nullable = true)
 |-- CO_FINANCIAMENTO: string (nullable = true)
 |-- CO_RUBRICA: string (nullable = true)
 |-- QT_TEMPO_PERMANENCIA: string (nullable = true)
 |-- DT_COMPETENCIA: string (nullable = true)
```

```
CID:
root
 |-- CO_CID: string (nullable = true)
 |-- NO_CID: string (nullable = true)
 |-- CO_CATEG: string (nullable = true)
 |-- NO_CATEG: string (nullable = true)
 |-- TP_AGRAVO: string (nullable = true)
 |-- TP_SEXO: string (nullable = true)
 |-- TP_ESTADIO: string (nullable = true)
 |-- VL_CAMPOS_IRRADIADOS: string (nullable = true)

ANO_MES:
root
 |-- PA_CMP: string (nullable = true)
 |-- ANO: string (nullable = true)
 |-- MES: string (nullable = true)
 |-- MES_NOME: string (nullable = true)
 |-- MES_ABREV: string (nullable = true)
 |-- TRIMESTRE: string (nullable = true)
 |-- SEMESTRE: string (nullable = true)

CBOCOD:
root
 |-- PA_CBOCOD: string (nullable = true)
 |-- CBOCOD_DESC: string (nullable = true)

TPUPS:
root
 |-- PA_TPUPS: string (nullable = true)
 |-- TPUPS_DESC: string (nullable = true)

CATEND:
root
 |-- PA_CATEND: string (nullable = true)
 |-- CATEND_DESC: string (nullable = true)

DOCORIG:
root
 |-- PA_DOCORIG: string (nullable = true)
 |-- DOCORIG_DESC: string (nullable = true)

SEXO:
root
 |-- PA_SEXO: string (nullable = true)
 |-- SEXO_DESC: string (nullable = true)

RACA_COR:
```

```
root
 |-- PA_RACACOR: string (nullable = true)
 |-- RACACOR_DESC: string (nullable = true)

MOTSAI:
root
 |-- PA_MOTSAI: string (nullable = true)
 |-- MOTSAI_DESC: string (nullable = true)
```

1.3 Exemplos de Consultas Analíticas

O objetivo dessa seção é mostrar alguns exemplos de utilização do modelo dimensional do nosso data warehouse.

Aqui, serão realizadas algumas consultas respondendo perguntas hipotéticas sobre a tabela de fatos.

1.3.1 Psicoterapia em Unidades Básicas de Saúde para recorte populacional e geográfico específico

Qual é a quantidade de procedimentos e valores aprovados pelo SUS para procedimentos **Psicoterápicos**, durantes os anos de 2018 a 2020, realizados em Unidades Básicas de Saúde, em pacientes Pretos e do sexo Masculino, agrupados por Região do País, Ano e Categoria do CID Primário?

[10]:
```
sia_df \
    .join(municipios_df.select('PA_UFMUN', 'REGIAO_SIGLA'), on='PA_UFMUN') \
    .join(ano_mes_df.select('PA_CMP', 'ANO'), on='PA_CMP') \
    .join(cnes_df.select('PA_CODUNI', 'TP_UNIDADE_DESC'), on='PA_CODUNI') \
    .join(sigtap_proced_df.select('PA_PROC_ID', 'NO_PROCEDIMENTO',␣
↪'DT_COMPETENCIA'), on=[sia_df['PA_PROC_ID'] ==␣
↪sigtap_proced_df['PA_PROC_ID'],
                                                                    ␣
↪          sia_df['PA_CMP'] == sigtap_proced_df['DT_COMPETENCIA']]) \
    .join(cid_df.select('CO_CID', 'NO_CID', 'CO_CATEG', 'NO_CATEG'),␣
↪on=sia_df['PA_CIDPRI'] == cid_df['CO_CID']) \
    .join(sexo_df, on='PA_SEXO') \
    .join(raca_cor_df, on='PA_RACACOR') \
    .where('TP_UNIDADE_DESC LIKE "%UNIDADE BASICA%"') \
    .where('NO_PROCEDIMENTO LIKE "%PSICOTERAPIA%"') \
    .where('ANO in ("2018", "2019", "2020")') \
    .where('SEXO_DESC = "Masculino"') \
    .where('RACACOR_DESC = "PRETA"') \
    .select('REGIAO_SIGLA', 'ANO', 'CO_CATEG', 'NO_CATEG', 'PA_QTDAPR',␣
↪'PA_VALAPR' ) \
    .groupBy('REGIAO_SIGLA', 'ANO', 'CO_CATEG', 'NO_CATEG' ) \
    .agg({'PA_QTDAPR':'sum', 'PA_VALAPR':'sum'}) \
```

```
    .sort('REGIAO_SIGLA', 'ANO', 'sum(PA_VALAPR)', ascending=[True, True,␣
 ↪False]) \
    .withColumn('sum(PA_VALAPR)', F.format_number('sum(PA_VALAPR)', 2)) \
    .withColumnRenamed('sum(PA_QTDAPR)', 'PROCEDIMENTOS') \
    .withColumnRenamed('sum(PA_VALAPR)', 'VALOR') \
    .show(60, truncate=False)
```

```
+------------+----+--------+--------------------------------------------------
----------+-------------+------+
|REGIAO_SIGLA|ANO |CO_CATEG|NO_CATEG
|PROCEDIMENTOS|VALOR |
+------------+----+--------+--------------------------------------------------
----------+-------------+------+
|NE          |2018|F20     |Esquizofrenia
|86           |219.30|
|NE          |2018|F32     |Episodios depressivos
|13           |33.15 |
|NE          |2019|F20     |Esquizofrenia
|13           |33.15 |
|NE          |2019|F41     |Transtornos ansiosos
|1            |2.55  |
|NE          |2019|F91     |Disturbios de conduta
|1            |2.55  |
|NE          |2020|F60     |Transtornos especificos da personalidade
|16           |40.80 |
|NE          |2020|F84     |Transtornos globais do desenvolvimento
|4            |10.20 |
|NE          |2020|F41     |Transtornos ansiosos
|0            |0.00  |
|S           |2020|F99     |Transtorno mental não especificado em outra parte
|4            |10.20 |
|SE          |2018|F91     |Disturbios de conduta
|15           |38.25 |
|SE          |2018|Z81     |História familiar de transtornos mentais e
comportamentais    |9            |22.95 |
|SE          |2018|F89     |Transtorno do desenvolvimento psicológico não
especificado    |4            |10.20 |
|SE          |2018|F90     |Transtornos hipercinéticos
|3            |7.65  |
|SE          |2018|E66     |Obesidade
|3            |7.65  |
|SE          |2018|F33     |Transtorno depressivo recorrente
|2            |5.10  |
|SE          |2018|R48     |Outras disfunções simbolicas não classificadas em
outra parte |2           |5.10  |
|SE          |2018|F32     |Episodios depressivos
|2            |5.10  |
|SE          |2018|F48     |Transtornos neuróticos
```

```
|2            |5.10  |
|SE           |2018|F20     |Esquizofrenia
|1            |2.55  |
|SE           |2018|F29     |Psicose não-orgânica não especificada
|1            |2.55  |
|SE           |2019|F60     |Transtornos especificos da personalidade
|34           |86.70 |
|SE           |2019|F91     |Disturbios de conduta
|15           |38.25 |
|SE           |2019|T74     |Síndromes de maus tratos
|13           |33.15 |
|SE           |2019|F29     |Psicose não-orgânica não especificada
|13           |33.15 |
|SE           |2019|F33     |Transtorno depressivo recorrente
|8            |20.40 |
|SE           |2019|F89     |Transtorno do desenvolvimento psicológico não
especificado    |5            |12.75 |
|SE           |2019|Z81     |História familiar de transtornos mentais e
comportamentais    |5            |12.75 |
|SE           |2019|F70     |Retardo mental leve
|3            |7.65  |
|SE           |2019|F90     |Transtornos hipercinéticos
|2            |5.10  |
|SE           |2019|F10     |Transtornos mentais e comportamentais devidos ao uso
de álcool|2            |5.10  |
|SE           |2019|E70     |Disturbios do metabolismo de aminoácidos aromáticos
|1            |2.55  |
|SE           |2019|R69     |Causas desconhecidas e não especificadas de
morbidade         |1            |2.55  |
|SE           |2019|Z56     |Problemas relacionados com o emprego e com o
desemprego       |1            |2.55  |
|SE           |2020|F33     |Transtorno depressivo recorrente
|22           |56.10 |
|SE           |2020|F20     |Esquizofrenia
|10           |25.50 |
|SE           |2020|F91     |Disturbios de conduta
|6            |15.30 |
|SE           |2020|F60     |Transtornos especificos da personalidade
|6            |15.30 |
|SE           |2020|F29     |Psicose não-orgânica não especificada
|3            |7.65  |
|SE           |2020|F32     |Episodios depressivos
|1            |2.55  |
|SE           |2020|F70     |Retardo mental leve
|1            |2.55  |
|SE           |2020|F71     |Retardo mental moderado
|1            |2.55  |
+-------------+----+--------+-------------------------------------------------
```

```
----------+-------------+------+
```

1.3.2 Procedimentos e Doenças que mais gastam recursos do SUS na cidade de Salto-SP em atendimentos ambulatoriais

Quais procedimentos mais demandaram recursos financeiros do SUS na cidade de Salto-SP no ano de 2020?

[11]:
```
sia_df \
    .join(municipios_df.select('PA_UFMUN', 'NOME', 'UF'), on='PA_UFMUN') \
    .join(ano_mes_df.select('PA_CMP', 'ANO'), on='PA_CMP') \
    .join(cid_df.select('CO_CID', 'NO_CID', 'CO_CATEG', 'NO_CATEG'),␣
 ↪on=sia_df['PA_CIDPRI'] == cid_df['CO_CID']) \
    .join(sigtap_proced_df.select('PA_PROC_ID', 'NO_PROCEDIMENTO',␣
 ↪'DT_COMPETENCIA'), on=[sia_df['PA_PROC_ID'] ==␣
 ↪sigtap_proced_df['PA_PROC_ID'],
                                                                    ␣
 ↪          sia_df['PA_CMP'] == sigtap_proced_df['DT_COMPETENCIA']]) \
    .where('ANO = "2020"') \
    .where('NOME ILIKE "SALTO" AND UF = "SP"') \
    .select('NO_PROCEDIMENTO', 'PA_QTDAPR', 'PA_VALAPR') \
    .groupBy('NO_PROCEDIMENTO') \
    .agg({'PA_QTDAPR':'sum', 'PA_VALAPR':'sum'}) \
    .sort('sum(PA_VALAPR)', ascending=False) \
    .withColumn('sum(PA_VALAPR)', F.format_number('sum(PA_VALAPR)', 2)) \
    .withColumnRenamed('sum(PA_QTDAPR)', 'PROCEDIMENTOS') \
    .withColumnRenamed('sum(PA_VALAPR)', 'VALOR') \
    .show(truncate=False)
```

```
+-------------------------------------------------------------------------------
-----------------------------------+-------------+----------+
|NO_PROCEDIMENTO
|PROCEDIMENTOS|VALOR     |
+-------------------------------------------------------------------------------
-----------------------------------+-------------+----------+
|TOMOGRAFIA COMPUTADORIZADA DE TORAX
|1750         |238,717.51|
|ATENDIMENTO FISIOTERAPÊUTICO NAS ALTERAÇÕES MOTORAS
|20799        |97,131.33 |
|TOMOGRAFIA COMPUTADORIZADA DO CRANIO
|585          |57,002.40 |
|EXAME ANATOMO-PATOLÓGICO PARA CONGELAMENTO / PARAFINA POR PEÇA CIRURGICA OU POR
BIOPSIA (EXCETO COLO UTERINO E MAMA)|1686         |40,464.00 |
|TOMOGRAFIA COMPUTADORIZADA DE PELVE / BACIA / ABDOMEN INFERIOR
|239          |33,132.57 |
|TOMOGRAFIA COMPUTADORIZADA DE ABDOMEN SUPERIOR
|237          |32,855.31 |
```

```
|ACOMPANHAMENTO NEUROPSICOLÓGICO DE PACIENTE EM REABILITAÇÃO
|1317          |23,271.39 |
|ULTRASSONOGRAFIA DE ABDOMEN TOTAL
|610           |23,149.50 |
|ATENDIMENTO / ACOMPANHAMENTO DE PACIENTE EM REABILITACAO DO DESENVOLVIMENTO
NEUROPSICOMOTOR                          |1149          |20,302.83 |
|COLONOSCOPIA (COLOSCOPIA)
|176           |19,828.16 |
|ESOFAGOGASTRODUODENOSCOPIA
|292           |14,062.72 |
|ULTRASSONOGRAFIA OBSTETRICA
|574           |13,890.80 |
|IMUNOHISTOQUIMICA DE NEOPLASIAS MALIGNAS (POR MARCADOR)
|120           |11,040.00 |
|TOMOGRAFIA COMPUTADORIZADA DE COLUNA LOMBO-SACRA C/ OU S/ CONTRASTE
|93            |9,402.30  |
|ULTRASSONOGRAFIA MAMARIA BILATERAL
|333           |8,058.60  |
|ULTRASSONOGRAFIA DE APARELHO URINÁRIO
|266           |6,437.20  |
|ATENDIMENTO FISIOTERAPÊUTICO EM PACIENTES COM DISTÚRBIOS NEURO-CINÉTICO-
FUNCIONAIS SEM COMPLICAÇÕES SISTÊMICAS      |965           |4,506.55  |
|TOMOGRAFIA COMPUTADORIZADA DE COLUNA CERVICAL C/ OU S/ CONTRASTE
|46            |3,990.96  |
|RESSONANCIA MAGNETICA DE COLUNA LOMBO-SACRA
|12            |3,225.00  |
|RESSONANCIA MAGNETICA DE MEMBRO INFERIOR (UNILATERAL)
|11            |2,956.25  |
+-------------------------------------------------------------------------------
-----------------------------------+-------------+----------+
only showing top 20 rows
```

De acordo com o resultado da consulta acima, é possível perceber que procedimentos de Tomografia Computadorizada são os que mais demandaram recursos financeiros do SUS para Salto-SP, em 2020.

Quais CIDs (Classificação Internacional de Doenças) levam à realização de procedimentos de tomografica computadorizada em Salto-SP? A consulta abaixo responde essa pergunta.

[12]:
```
sia_df \
    .join(municipios_df.select('PA_UFMUN', 'NOME', 'UF'), on='PA_UFMUN') \
    .join(ano_mes_df.select('PA_CMP', 'ANO'), on='PA_CMP') \
    .join(cid_df.select('CO_CID', 'NO_CID', 'CO_CATEG', 'NO_CATEG'),␣
 ↪on=sia_df['PA_CIDPRI'] == cid_df['CO_CID']) \
    .join(sigtap_proced_df.select('PA_PROC_ID', 'NO_PROCEDIMENTO',␣
 ↪'DT_COMPETENCIA'), on=[sia_df['PA_PROC_ID'] ==␣
 ↪sigtap_proced_df['PA_PROC_ID'],
```

```
          sia_df['PA_CMP'] == sigtap_proced_df['DT_COMPETENCIA']]) \
    .where('ANO = "2020"') \
    .where('NOME ILIKE "salto" AND UF = "SP"') \
    .where('NO_PROCEDIMENTO ILIKE "%tomografia%computadorizada%"') \
    .select('CO_CATEG', 'CO_CID', 'NO_CID', 'PA_QTDAPR', 'PA_VALAPR') \
    .groupBy('CO_CATEG', 'CO_CID', 'NO_CID') \
    .agg({'PA_QTDAPR':'sum', 'PA_VALAPR':'sum'}) \
    .sort('sum(PA_VALAPR)', ascending=False) \
    .withColumn('sum(PA_VALAPR)', F.format_number('sum(PA_VALAPR)', 2)) \
    .withColumnRenamed('sum(PA_QTDAPR)', 'PROCEDIMENTOS') \
    .withColumnRenamed('sum(PA_VALAPR)', 'VALOR') \
    .show(truncate=False)
```

```
+--------+------+-------------------------------------------------------------------
----------------+------------+----------+
|CO_CATEG|CO_CID|NO_CID
|PROCEDIMENTOS|VALOR     |
+--------+------+-------------------------------------------------------------------
----------------+------------+----------+
|R52     |R529  |Dor não especificada
|2899         |366,800.78|
|R10     |R104  |Outras dores abdominais e as não especificadas
|59           |7,980.39  |
|I64     |I64   |Acidente vascular cerebral, não especificado como hemorrágico
ou isquêmico      |74           |7,210.56  |
|M79     |M796  |Dor em membro
|17           |1,683.35  |
|M51     |M510  |Transtornos de discos lombares e de outros discos
intervertebrais com mielopatia|1            |101.10    |
|S52     |S520  |Fratura da extremidade superior do cúbito [ulna]
|1            |86.75     |
|T14     |T141  |Ferimento de região não especificada do corpo
|1            |86.75     |
+--------+------+-------------------------------------------------------------------
----------------+------------+----------+
```

Em segundo lugar, o procedimento **Atendimento fisioterapêutico nas alterações motoras** também gasta recursos consideráveis do SUS em Salto-SP. Quais CIDs fazem uso desse procedimento? A consulta abaixo responde a essa pergunta.

[13]:
```
sia_df \
    .join(municipios_df.select('PA_UFMUN', 'NOME', 'UF'), on='PA_UFMUN') \
    .join(ano_mes_df.select('PA_CMP', 'ANO'), on='PA_CMP') \
    .join(cid_df.select('CO_CID', 'NO_CID', 'CO_CATEG', 'NO_CATEG'),
 ↪on=sia_df['PA_CIDPRI'] == cid_df['CO_CID']) \
```

```
   .join(sigtap_proced_df.select('PA_PROC_ID', 'NO_PROCEDIMENTO','DT_COMPETENCIA'), on=[sia_df['PA_PROC_ID'] ==sigtap_proced_df['PA_PROC_ID'],
                                                                                        
          sia_df['PA_CMP'] == sigtap_proced_df['DT_COMPETENCIA']]) \
   .where('ANO = "2020"') \
   .where('NOME ILIKE "salto" AND UF = "SP"') \
   .where('NO_PROCEDIMENTO ILIKE "%ATENDIMENTO FISIOTERAPÊUTICO NAS ALTERAÇÕES MOTORAS%"') \
   .select('CO_CATEG', 'CO_CID', 'NO_CID', 'PA_QTDAPR', 'PA_VALAPR') \
   .groupBy('CO_CATEG', 'CO_CID', 'NO_CID') \
   .agg({'PA_QTDAPR':'sum', 'PA_VALAPR':'sum'}) \
   .sort('sum(PA_VALAPR)', ascending=False) \
   .withColumn('sum(PA_VALAPR)', F.format_number('sum(PA_VALAPR)', 2)) \
   .withColumnRenamed('sum(PA_QTDAPR)', 'PROCEDIMENTOS') \
   .withColumnRenamed('sum(PA_VALAPR)', 'VALOR') \
   .show(truncate=False)
```

```
+--------+------+-------------------------------------------------------------+----
---------+---------+
|CO_CATEG|CO_CID|NO_CID
|PROCEDIMENTOS|VALOR    |
+--------+------+-------------------------------------------------------------+----
---------+---------+
|M96     |M969  |Transtorno osteomuscular não especificado pós-
procedimento|10650        |49,735.50|
|M54     |M544  |Lumbago com ciática                                          |8929
|41,698.43|
|M25     |M255  |Dor articular                                                |861
|4,020.87 |
|M65     |M654  |Tenossinovite estilóide radial [de quervain]                 |296
|1,382.32 |
|G81     |G811  |Hemiplegia espástica                                         |25
|116.75   |
|M65     |M659  |Sinovite e tenossinovite não especificadas                   |19
|88.73    |
|M25     |M256  |Rigidez articular não classificada em outra parte            |18
|84.06    |
|M75     |M754  |Síndrome de colisão do ombro                                 |1
|4.67     |
+--------+------+-------------------------------------------------------------+----
---------+---------+
```

Qual é o perfil dos pacientes que mais utilizaram o procedimento **Atendimento fisioterapêutico nas alterações motoras** em Salto-SP em 2020?

[14]:
```
sia_df \
    .join(municipios_df.select('PA_UFMUN', 'NOME', 'UF'), on='PA_UFMUN') \
    .join(ano_mes_df.select('PA_CMP', 'ANO'), on='PA_CMP') \
    .join(sigtap_proced_df.select('PA_PROC_ID', 'NO_PROCEDIMENTO',␣
 ↪'DT_COMPETENCIA'), on=[sia_df['PA_PROC_ID'] ==␣
 ↪sigtap_proced_df['PA_PROC_ID'],
                                                                    ␣
 ↪          sia_df['PA_CMP'] == sigtap_proced_df['DT_COMPETENCIA']]) \
    .join(sexo_df, on='PA_SEXO') \
    .join(raca_cor_df, on='PA_RACACOR') \
    .where('ANO = "2020"') \
    .where('NOME ILIKE "salto" AND UF = "SP"') \
    .where('NO_PROCEDIMENTO ILIKE "%ATENDIMENTO FISIOTERAPÊUTICO NAS ALTERAÇÕES␣
 ↪MOTORAS%"') \
    .select('SEXO_DESC', 'RACACOR_DESC', 'PA_IDADE', 'PA_QTDAPR', 'PA_VALAPR') \
    .groupBy('SEXO_DESC', 'RACACOR_DESC') \
    .agg({'PA_IDADE':'avg', 'PA_QTDAPR':'sum', 'PA_VALAPR':'sum'}) \
    .sort('sum(PA_VALAPR)', ascending=False) \
    .withColumn('sum(PA_VALAPR)', F.format_number('sum(PA_VALAPR)', 2)) \
    .withColumn('avg(PA_IDADE)', F.round('avg(PA_IDADE)', 1)) \
    .withColumnRenamed('avg(PA_IDADE)', 'IDADE_MEDIA') \
    .withColumnRenamed('sum(PA_QTDAPR)', 'PROCEDIMENTOS') \
    .withColumnRenamed('sum(PA_VALAPR)', 'VALOR') \
    .show(truncate=False)
```

```
+---------+--------------+-------------+---------+-----------+
|SEXO_DESC|RACACOR_DESC  |PROCEDIMENTOS|VALOR    |IDADE_MEDIA|
+---------+--------------+-------------+---------+-----------+
|Feminino |BRANCA        |9429         |44,033.43|53.9       |
|Masculino|BRANCA        |5857         |27,352.19|47.2       |
|Feminino |PARDA         |2208         |10,311.36|51.8       |
|Masculino|PARDA         |1588         |7,415.96 |47.8       |
|Feminino |PRETA         |1011         |4,721.37 |54.8       |
|Masculino|PRETA         |606          |2,830.02 |47.3       |
|Feminino |AMARELA       |61           |284.87   |48.4       |
|Masculino|SEM INFORMAÇÃO|26           |121.42   |55.5       |
|Feminino |SEM INFORMAÇÃO|12           |56.04    |58.3       |
|Masculino|AMARELA       |1            |4.67     |59.0       |
+---------+--------------+-------------+---------+-----------+
```

1.4 Encerramento da sessão Spark

[15]:
```
spark.stop()
```

[ ]:
```

```

ANEXOS

ANEXO A – BASES DE DADOS DO SUS FORNECIDAS PELO DESAFIO DATATHON

Datathon

Sobre os dados

Todos os dados disponibilizados são públicos e disponíveis nos sites oficiais do governo. Os dados estão divididos nas seguintes categorias:

- Dados principais
  Dados coletados e disponibilizados para análise de acordo com o desafio proposto.

- Dados de apoio
  Com o objetivo de facilitar a análise dos dados principais, os dados de apoio contêm entre outras informações a descrição de diversos campos informados nos dados principais em formato numérico. Por exemplo:

  Dado principal > UF = 27
  Dado de apoio > 27 = AL

- Documentação
  Detalhamento da origem dos dados, dicionários e descrições detalhadas disponibilizadas pelos portais oficiais de disponibilização dos dados.
  Adicionamos também um notebook python com alguma orientação de uma leitura básica de um arquivo parket.

ATENÇÃO

Devido ao volume os dados estão disponibilizados em parquet. Não recomendamos a conversão para outros formatos, como o csv por exemplo, os dados totais em csv podem ultrapassar 500GB.

Para facilitar a análise os datasets de maior volume foram particionados por ano ou UF, o detalhamento está descrito neste documento.

Veja abaixo a descrição de cada arquivo disponível para download neste repositório.

Dados principais

Arquivo	Descrição
tb_sih_rd.zip	Sistema de Informações Hospitalares do SUS RD – AIH Reduzida Dados totais em arquivo parquet particionado.
tb_sia_pa.zip	Sistema de Informações Ambulatoriais do SUS PA – Produção Ambulatorial Dados totais em arquivo parquet particionado.
tb_sia_pa_partitioned.zip	Sistema de Informações Ambulatoriais do SUS PA – Produção Ambulatorial

	Dados particionados por ano e estado em arquivo parquet.
Inss_beneficios.zip	Instituto Nacional de Seguro Social – INSS Benefícios concedidos e indeferidos Dados particionados por status em arquivo parquet.

Dados de apoio

Arquivo	Descrição
tb_cnes_estabelecimentos.zip	CNES – Cadastro Nacional de Estabelecimentos de Saúde Dados totais em arquivo parquet particionado.
tb_cnes_estabelecimentos_partitionated.zip	CNES – Cadastro Nacional de Estabelecimentos de Saúde Dados particionados por estado em arquivo parquet.
tb_ibge_municipio.zip	Instituto Brasileiro de Geografia e Estatística Relação de municípios Dados totais em arquivo parquet particionado.
tb_ibge_uf.zip	Instituto Brasileiro de Geografia e Estatística Relação de estados Dados totais em arquivo parquet particionado.
tb_sigtap_cid.zip	Sistema de Gerenciamento da Tabela de Procedimentos, Medicamentos e OPM do SUS Relação de CID Dados totais em arquivo parquet particionado.
tb_sigtap_procedimento.zip	Sistema de Gerenciamento da Tabela de Procedimentos, Medicamentos e OPM do SUS Relação de procedimentos Dados totais em arquivo parquet particionado.

Documentação

Arquivo	Descrição
INSS_beneficios.pdf	Informações sobre origem dos dados e transformações realizadas.
IT_CNES.pdf	Documento disponibilizado no datasus.

IT_SIHSUS.pdf	Documento disponibilizado no datasus.
SIASUS.pdf	Documento disponibilizado no datasus.
read_parquet-to-pandasdf.ipynb	Notebook em python de exemplo para leitura dos arquivos parquet em dataframe pandas.
ler_parquet.R	Notebook em R de exemplo para leitura dos arquivos parquet.
exemplo.parquet exemplo_parquet.Rmd	Arquivos de exemplo utilizados no notebook em R.