UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

RAFAEL JÚNIOR RIBEIRO

Uma proposta de extração, transformação, carga e visualização para os dados do Censo Escolar

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientadora: Profa. Dra. Renata Galante

Porto Alegre
2023

AGRADECIMENTOS

Agradeço a minha família, principalmente a minha mãe Romilda, que me apoiou desde o pré-vestibular, e as políticas públicas para inclusão e permanência de cidadãos pobres no Ensino Superior, que me forneceram moradia, alimentação e transporte, me ajudando principalmente na etapa inicial no curso.

RESUMO

Neste trabalho é discutido a importância dos dados abertos para a sociedade e a ciência, enfatizando sua capacidade de promover a transparência, prestação de contas e tomada de decisões baseadas em dados. Para ilustrar esse ponto, é utilizado o Censo Escolar da Educação Básica como exemplo, por meio da elaboração de um projeto prático que engloba todas as etapas necessárias para o seu uso eficiente em mineração de dados, com o objetivo final de facilitar a busca de soluções para desafios na educação básica brasileira, e, assim, contribuir para sua melhoria e desenvolvimento a longo prazo. Foi elaborada a extração e transformação dos dados de 2016 até 2022. As informações coletadas são resumidas e apresentadas por meio de gráficos que apresentam informações sobre acesso de escola a serviços básicos (água, energia elétrica, esgoto, etc), quantidade de matrículas e escolas, assim como indicadores educacionais.

Palavras-chave: Plataforma Web. educação. visualização. etl. dados abertos. base de dados. censo escolar.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ETL	*Extract, transform and load*
ELT	*Extract, load and transform*
ETLT	*Extract, transform, load and transform*
SQL	*Structured Query Language*
INEP	*Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira*
MEC	*Ministério da Educação*
CSV	*Comma-separated values*
AFD	*Adequação da Formação Docente*
IED	*Índice de Esforço Docente*
ATU	*Média de Alunos por Turma*
HAD	*Média de Horas-aula diária*
DSU	*Percentual de Docentes com Curso Superior*
TDI	*Taxa de Distorção Idade-série*
TAP	*Taxa de Aprovação*
TRP	*Taxa de Reprovação*
TAB	*Taxa de Abandono*
IDEB	*Índice de Desenvolvimento da Educação Básica*
SGBD	*Sistema de Gerenciamento de Banco de Dados*
OLAP	*On-Line Analytical Processing*
IFRN	*Instituto Federal do Rio Grande do Norte*

SUMÁRIO

1 INTRODUÇÃO

Os dados abertos são importantes para a sociedade e a ciência por diversas razões. Em primeiro lugar, eles promovem a transparência e a prestação de contas das instituições governamentais e não governamentais, permitindo que os cidadãos e os pesquisadores possam acessar e analisar informações relevantes para tomar decisões orientadas a dados e monitorar o desempenho das políticas públicas. Além disso, os dados abertos também podem ser utilizados para gerar insights e soluções inovadoras para problemas sociais e ambientais, por meio de análises de big data e inteligência artificial. Segundo um estudo da McKinsey Global Institute[1], o uso de dados abertos pode gerar um valor econômico de até US$ 3 trilhões por ano em todo o mundo, ao permitir a criação de novos serviços e produtos, melhorar a eficiência dos processos produtivos e reduzir os custos operacionais. Além disso, a publicação de dados abertos também pode fortalecer a colaboração entre instituições e comunidades, ao permitir o compartilhamento de conhecimento e o desenvolvimento de redes de cooperação.

No Brasil, ainda que a transparência governamental tenha sofrido desgastes nos últimos anos [Lopes, Meyer e Linhares 2021], numa escala de tempo maior houve avanços na disponibilização de dados. Na esfera federal, se destaca o Portal da Transparência, DataSUS e o portal de dados do INEP, onde são disponibilizados diversos conjuntos de dados relacionados com a educação, sendo um deles o Censo Escolar. O Censo Escolar é a principal pesquisa estatística da educação básica no Brasil, coordenada pelo Inep em colaboração com as secretarias estaduais e municipais de educação. Abrange todas as escolas públicas e privadas, incluindo as diferentes etapas e modalidades da educação básica e profissional. A coleta de dados é declaratória e acontece em duas etapas: matrícula inicial e situação do aluno. A realização do Censo é descentralizada e envolve a colaboração entre União, estados e municípios, cada um com suas atribuições definidas pela Portaria do MEC [Brasil, 2022]. É uma ferramenta valiosa para compreender a

[1] Disponível em <https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/open-data-unlocking-innovation-and-performance-with-liquid-information>. Acesso em: 2 abr. 2023

situação educacional do país, das escolas e para acompanhar políticas públicas, assim como para definir o repasse de recursos públicos.

O propósito deste trabalho é estabelecer uma rotina eficiente de extração, transformação, carga e visualização para o Censo Escolar, com o objetivo final de facilitar a busca de soluções para desafios na educação básica brasileira, e assim, contribuir para sua melhoria e desenvolvimento a longo prazo. A principal tecnologia utilizada nesse trabalho é a linguagem de programação Python, amplamente utilizada na área de dados, oferecendo uma grande variedade de bibliotecas e ferramentas para realizar tarefas relacionadas à coleta, processamento, modelagem e análise de dados. Para o processamento de dados foram utilizadas 2 bibliotecas do Python: Pandas para transformação e DuckDB para análise colunar. Para criação da aplicação de dados é utilizado Streamlit, onde são realizados experimentos que mostram a evolução da quantidade de escolas e matrículas ao longo dos anos, e situação das escolas no acesso a serviços básicos e indicadores educacionais.

O restante do texto está dividido da seguinte forma: no Capítulo 2 estão resumidos os conceitos e tecnologias necessárias para o entendimento do trabalho, assim como uma descrição das bases de dados utilizadas. No Capítulo 3 são apresentados trabalhos acadêmicos relacionados, fazendo comparações com este trabalho. O Capítulo 4 descreve sobre seu desenvolvimento e implementação, com o Capítulo 5 apresentado os resultados obtidos na aplicação de dados, concluindo no Capítulo 6, que sumariza o projeto desenvolvido, sua contruibuição para a pesquisa na educação, assim como trabalhos futuros.

2 CONCEITOS, DADOS E TECNOLOGIAS

Este Capítulo apresenta as tecnologias, aplicações, técnicas e conceitos que foram utilizados no desenvolvimento deste trabalho, que abrangem diversas áreas abordadas no curso de Ciência da Computação como Bancos de Dados, Engenharia de Software, Visualização de Dados, entre outros.

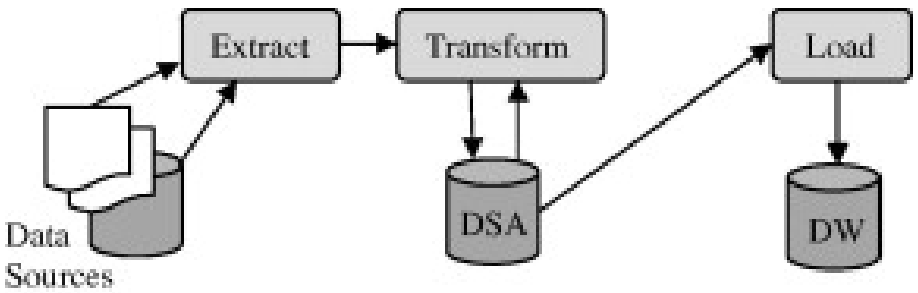
2.1 Conceitos

Esta seção aborda os conceitos necessários para um bom entendimento do trabalho.

2.1.1 ETL

Extract, Transform, Load, sigla em Ingês para a Extração, Transformação e Carga (ETL) é um processo crucial no gerenciamento e integração de dados, amplamente utilizado em projetos de Data Warehouse, Business Intelligence e Big Data. O ETL é responsável por coletar dados de várias fontes, aplicar transformações para melhorar a qualidade dos dados e carregá-los em um destino final, como um banco de dados ou armazém de dados (EL-SAPPAGH et al., 2011). A Figura 2.1 esquematiza de forma genérica um processo de ETL.

Figura 2.1: Framework genérica para um processo de ETL.



Fonte: El-Sappagh et al. (2011, p.92).

A etapa de extração envolve a coleta de dados brutos de múltiplas fontes

heterogêneas, como arquivos de texto, planilhas, bancos de dados relacionais e não-relacionais, sistemas legados, APIs, entre outros. O objetivo é extrair dados de forma consistente e eficiente, considerando questões como formato, volume e velocidade.

A etapa de transformação é responsável por processar e melhorar os dados extraídos. Isso inclui a limpeza, padronização, enriquecimento, agregação e integração dos dados. Algumas das transformações mais comuns incluem remoção de duplicatas, conversão de tipos de dados, aplicação de regras de negócio, normalização e criação de novos atributos derivados. Esta fase é crucial para garantir a qualidade dos dados e sua adequação às necessidades analíticas ou de negócio.

A fase final do processo ETL é o carregamento dos dados transformados no destino final, como um armazém de dados, um data lake ou um banco de dados analítico. A carga pode ser feita de maneira incremental, atualizando apenas os registros modificados, ou em lotes, onde todos os dados são carregados de uma vez. A escolha da estratégia de carga depende das necessidades de negócio e da arquitetura do sistema.

Existem variações do ETL, como o ELT e o ETLT que surgiram como resposta a diferentes necessidades e contextos em projetos de gerenciamento e integração de dados (VASSILIADIS; SIMITSIS, 2009). No ELT, em vez de transformar os dados antes de carregá-los no armazém de dados, os dados brutos são carregados diretamente no destino e realiza as transformações dentro do próprio armazém. Esta abordagem tira proveito do poder de processamento dos modernos sistemas de armazenamento de dados, como os *data warehouses* em nuvem, que são projetados para lidar com grandes volumes de dados e realizar transformações complexas. O ELT pode oferecer melhor desempenho e escalabilidade em comparação com o ETL tradicional em algumas situações. Além disso, fica mais fácil a transformação de dados para novas necessidade analíticas.

O ETLT é uma combinação das abordagens ETL e ELT, na qual algumas transformações são realizadas antes da carga (como no ETL) e outras são realizadas após a carga (como no ELT). Essa abordagem é útil quando se deseja realizar transformações iniciais para melhorar a qualidade dos dados ou reduzir o volume antes de carregá-los no destino, e posteriormente

realizar transformações mais complexas ou específicas do domínio no próprio armazém de dados.

2.1.2 SQL

SQL, ou Structured Query Language, é uma linguagem de programação usada para gerenciar bancos de dados relacionais (DATE, 1989). Com o SQL, é possível criar, modificar e consultar dados em SGBDs (Sistemas Gerenciadores de Bancos de Dados).

Os principais comandos do SQL incluem:

- SELECT: usado para selecionar dados de uma ou mais tabelas;
- INSERT: usado para inserir novos dados em uma tabela;
- UPDATE: usado para atualizar dados existentes em uma tabela;
- DELETE: usado para excluir dados de uma tabela;
- CREATE: usado para criar novas tabelas, índices e outros objetos no banco de dados;
- ALTER: usado para alterar a estrutura de uma tabela existente;
- DROP: usado para excluir uma tabela ou outro objeto do banco de dados.

O SQL também possui cláusulas, que permitem filtrar e ordenar dados, bem como agrupá-los e calcular funções agregadas. Algumas cláusulas comuns incluem:

- WHERE: usada para filtrar dados com base em uma condição;
- ORDER BY: usada para ordenar os resultados por uma ou mais colunas;
- GROUP BY: usada para agrupar os resultados com base em uma ou mais colunas;
- HAVING: usada para filtrar grupos com base em uma condição;
- JOIN: usada para combinar dados de duas ou mais tabelas com base em uma condição.

Há uma implentação padrão, regulada pela ANSI (American National Standards Institute), assim como dialetos próprios, cada um com suas próprias características e recursos específicos. Alguns exemplos incluem o MySQL, o PostgreSQL e o Oracle SQL.

SQL inicialmente era utilizado exclusivamente em SGBDs monolíticos, porém atualmente muitos sistemas de processamento de big data, como o Apache Hadoop e o Apache Spark (SILVA et al., 2016), oferecem suporte a consultas SQL, permitindo que os usuários escrevam consultas SQL para extrair informações de grandes conjuntos de dados armazenados em clusters distribuídos.

2.1.3 Particionamento de Dados

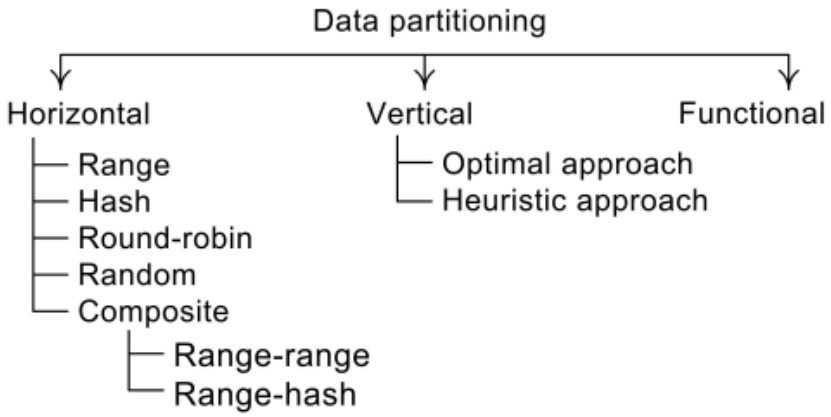
Atualmente, vivemos na era da informação, onde somos constantemente bombardeados por uma quantidade massiva de dados gerados de diversas fontes, como redes sociais, sensores, transações comerciais e até mesmo dispositivos pessoais. Essa crescente quantidade de dados traz consigo desafios específicos de gerenciamento, processamento e armazenamento. Um dos principais aspectos a serem considerados nesse contexto é o particionamento de dados (MAHMUD et al., 2020).

O particionamento de dados é uma técnica essencial para lidar com grandes volumes de informações, que visa dividir um conjunto de dados maior em partes menores e mais gerenciáveis. Essa abordagem oferece diversos benefícios, como melhor desempenho, escalabilidade e eficiência na recuperação e processamento de dados. Existem dois tipos principais de particionamento: horizontal e vertical, conforme resumido na Figura 2.2.

O particionamento horizontal envolve a divisão de um conjunto de dados em várias tabelas ou partições que contêm um subconjunto de linhas do conjunto original. Cada partição é tratada como uma entidade separada e pode ser distribuída em diferentes nós de processamento ou armazenamento. Isso melhora a eficiência na consulta e recuperação de dados, bem como a distribuição de carga.

Por outro lado, o particionamento vertical consiste em dividir uma tabela, armazenando diferentes colunas em diferentes tabelas ou partições. Isso é útil quando um conjunto de dados possui colunas que são acessadas com frequências diferentes ou quando algumas colunas contêm informações confidenciais que devem ser isoladas do restante dos dados.

Figura 2.2: Subtipos de particionamento.



Fonte: Mahmud et al. (2020, p. 88).

Um perigo associado ao particionamento de dados é a criação de partições desbalanceadas, o que ocorre quando uma ou mais partições contêm uma quantidade significativamente maior de dados do que outras. Isso pode levar a gargalos no desempenho e sobrecarga de recursos em determinados nós de processamento ou armazenamento.

Para evitar partições desbalanceadas, é importante seguir algumas práticas recomendadas. Em primeiro lugar, é crucial analisar o conjunto de dados e compreender como os dados são distribuídos antes de definir a estratégia de particionamento. Além disso, é importante escolher uma chave de particionamento adequada que permita uma distribuição uniforme dos dados entre as partições. Em alguns casos, pode ser necessário reavaliar e ajustar periodicamente a estratégia de particionamento para garantir que as partições continuem balanceadas à medida que novos dados são inseridos ou removidos do conjunto de dados.

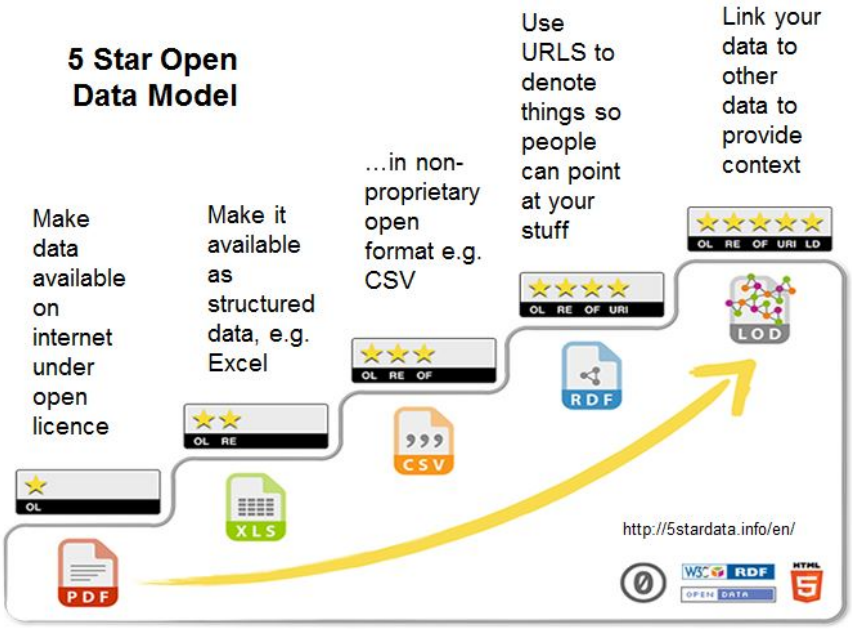
2.1.4 Dados Abertos

De acordo com a *Open Knowledge Foundation* (OPENDATAHANDBOOK.ORG, 2023), uma organização sem fins lucrativos que promove conhecimento livre, dados abertos - open data em Inglês - são dados que podem ser livremente usados, reutilizados e redistribuídos por qualquer pessoa - sujeitos, no máximo, à exigência de atribuição da fonte e

compartilhamento pelas mesmas regras.

Tim Berners-Lee, criador da World Wide Web e ativista de dados abertos, sugere uma classificação para a qualidade de dados abertos em 5 estrelas (BERNERS-LEE, 2006), conforme mostrado na Figura 2.3.

Figura 2.3: Classificação da qualidade de dados abertos.



Fonte: Site Five Star Data[1]

Os dados abertos governamentais são informações disponibilizadas pelos governos, sejam elas sobre políticas públicas, gastos públicos, estatísticas, entre outras. O objetivo é tornar essas informações acessíveis ao público, de forma a aumentar a transparência e a responsabilização dos governos perante a sociedade. O conceito de dados governamentais abertos é baseado na filosofia do código aberto (open source), que se fundamenta em três pilares conceituais: abertura, participação e colaboração (ISOTANI; BITTENCOURT, 2015).

Existem diversas iniciativas de dados abertos governamentais ao redor do mundo. No Brasil se destaca o Portal da Transparência, DataSUS e Dados.gov.br. Esses portais permitem que qualquer pessoa possa acessar e baixar os dados em diversos formatos, além de disponibilizarem ferramentas para análise e visualização desses dados.

Os benefícios dos dados abertos governamentais são muitos. Além de

[1]Disponível em <https://5stardata.info/en/>. Acesso em: 21 mar. 2023.

promover a transparência e a participação do cidadão, eles também podem ser usados para fins educacionais, de pesquisa e de desenvolvimento de novas tecnologias. Além disso, os dados abertos podem ajudar a melhorar a eficiência da gestão pública, ao permitir que os governos tenham uma visão mais clara dos problemas e das necessidades da sociedade.

No entanto, existem alguns desafios e preocupações relacionados aos dados abertos governamentais. Um dos principais é a privacidade dos dados pessoais dos cidadãos. É preciso garantir que essas informações sejam protegidas e que o seu uso seja restrito a fins legítimos. Além disso, também é importante garantir a qualidade e a integridade dos dados, para que eles possam ser usados de forma confiável (JANSSEN et al., 2012).

2.1.5 Censo Escolar

O Censo Escolar é uma pesquisa realizada anualmente pelo Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP), órgão vinculado ao Ministério da Educação (MEC) do Brasil. A pesquisa tem como objetivo coletar informações sobre as escolas, alunos, professores e demais profissionais da educação em todo o país.

O Censo Escolar é importante porque fornece dados precisos e atualizados sobre o sistema educacional brasileiro, permitindo que o governo e os pesquisadores entendam melhor as tendências e necessidades educacionais em todo o país. Com base nesses dados, o MEC pode tomar decisões mais informadas sobre como investir recursos e melhorar a qualidade da educação no país.

As informações coletadas no Censo Escolar incluem dados demográficos dos alunos, como idade, sexo, raça e etnia, bem como informações sobre o desempenho escolar, como notas e resultados de avaliações. Além disso, o Censo Escolar também coleta dados sobre a infraestrutura das escolas, como número de salas de aula, bibliotecas e laboratórios, bem como informações sobre os professores e outros profissionais da educação.

A participação no Censo Escolar é obrigatória para todas as escolas brasileiras, sejam elas públicas ou privadas. As escolas são responsáveis por

enviar as informações solicitadas pelo INEP por meio do Sistema Educacenso, que é um sistema online desenvolvido pelo instituto para coletar e processar os dados do censo.

O Censo Escolar fornece uma ampla gama de informações que permitem a criação de diversos indicadores educacionais relevantes. Esses indicadores podem ser usados para avaliar a qualidade da educação no Brasil, identificar tendências e necessidades educacionais, além de orientar políticas públicas e tomada de decisões na área da educação. Entre os indicadores estão o índice de desenvolvimento da educação básica (ldeb), índice de esforço docente, taxas de distorção idade-série, média de alunos por turma, etc.

Em 2022, houve uma mudança significativa na forma como os dados do Censo Escolar são disponibilizados. Anteriormente, esses dados eram divulgados por meio do portal do INEP, em arquivos em formato Excel compactados dentro de pastas também compactadas. Os dados eram normalizados onde, para haver uma visão adequada dos dados, era necessário fazer junções demoradas entre as tabelas. Isso dificultava a análise e a comparação dos dados por pesquisadores e demais interessados.

Embora a mudança na disponibilização dos dados tenha melhorado, ela também levantou algumas críticas por parte de especialistas em educação. Isso ocorreu porque, ao disponibilizar os dados de forma mais acessível e amigável, houve uma diminuição na granularidade dos dados, ou seja, na quantidade de informações detalhadas disponíveis em cada item do Censo. Antes a granularidade chegava ao nível mais básico, a matrícula do aluno, com informações também sobre docentes e diretores. Porém no novo modelo as informações individuais foram agregadas a nível de escola (LGPD-BRASIL, 2022).

De acordo com os críticos, a diminuição na granularidade dos dados pode prejudicar a análise e o entendimento de certas questões importantes do sistema educacional brasileiro, como as desigualdades regionais e socioeconômicas no acesso à educação, por exemplo. Além disso, há preocupações com a possibilidade de que a simplificação dos dados possa levar a uma visão simplista e superficial do sistema educacional, sem considerar a complexidade de questões importantes que afetam a qualidade

da educação no país.

No entanto, o INEP argumenta que a alteração foi para adequar a base de dados à Lei Geral de Proteção de Dados Pessoais (LGPD), e que a plataforma de dados abertos do Censo Escolar ainda oferece informações suficientes para uma análise detalhada do sistema educacional brasileiro, incluindo dados sobre a rede de ensino, os níveis e modalidades de ensino, as características dos alunos e professores, entre outros aspectos.

Aprovada em Abril de 2022 na Câmara dos Deputados e atualmente aguardando relatoria no Senado Federal, está o projeto de lei 454/22, que autoriza o poder público a compartilhar e a tornar públicos de forma completa os dados e microdados obtidos por meio do Censo Escolar e dos exames de avaliação dos estudantes (BRASIL, 2022).

2.2 Dados

Esta Seção discorre sobre as fontes das quais os dados utilizados para o desenvolvimento foram retirados.

2.2.1 Microdados

Os registros detalhados a nível de escola foram retirados do portal do INEP (BRASIL, 2023a). O período analisado vai de 2016 até 2022. Há 1608986 linhas com 370 atributos, com dados faltantes em 0.8% das células. Armezenado no formato CSV (comma-separated values) cada ano possui em média 200 MB. Transformados e salvos no formato Parquet, cada ano possui em média 30 MB de tamanho. Carregado totalmente na memória, o dataset transformado ocupa em torno de 1.5 GB. Para melhor performance nas consultas, o dataset foi salvo de forma desnormalizada, não sendo necessário fazer joins com outras tabelas. O problema da redundância de dados é minimizado com o uso do Parquet e seus algoritmos de compreensão e descompreensão. O dataset compõe informações sobre 4 grupos: matrículas, docentes, gestores e escolas. 42 colunas são categóricas nominais, 2 de tipo data, 210 booleano e 116 de tipo numérico. O prefixo de cada coluna ajuda e

entender qual é o domínio de valores possíveis para a coluna.

Tabela 2.1: Prefixos de colunas

Prefixo	Tipo	Descrição	Exemplo
NO	String	Nome de localidade geográfica	NO_REGIAO (nome da região geográfica) NO_MUNICIPIO (nome do município)
SG	String	Sigla	SG_UF (sigla do unidade federativa)
CO	String	Código identificador	CO_MESORREGIAO (código da messoregião) CO_ENTIDADE (código da entidade)
TP	String	Variável categórica nominal	TP_DEPENDENCIA (dependência administrativa) TP_CATEGORIA_ESCOLA_PRIVADA (categoria da escola privada)
DS	String	Endereço da escola	DS_ENDERECO (endereço) DS_COMPLEMENTO (complemento)
NU	String/Inteiro	Número	NU_ANO_CENSO (ano do censo) CO_CEP (CEP)
DT	Date	Data	DT_ANO_LETIVO_INICIO (Início do ano letivo) DT_ANO_LETIVO_TERMINO (Término do ano letivo)
IN	Booleano	2 valores possíveis: 0 para não e 1 para sim	IN_AGUA_REDE_PUBLICAL (abastecimento de água via rede pública) IN_AGUA_POCO_ARTESIANO (abastecimento de água via poço artesiano)
QT	Inteiro	Valores quantitativos discretos de até 4 digitos	QT_MAT_MED (Quantidade de matrículas no ensino médio) QT_DESKTOP_ALUNO (quantidade de computadores em uso pelos alunos)

Fonte: o autor.

2.2.2 Indicadores Educacionais

Calculados com base nos dados do Censo Escolar, o INEP disponibiliza um conjunto amplo de indicadores que possibilitam monitorar o desenvolvimento da educação brasileira (BRASIL, 2023b). Para esse trabalho, foram extraídos, de 2016 até 2022, agregados a nível municipal, estadual, regional e federal, os seguintes conjuntos:

- Adequação da Formação Docente: indicador da adequação da formação inicial dos docentes das escolas de educação básica brasileira, segundo as orientações legais (BRASIL, 2023c). O professores de cada disciplina são divididos em 5 grupos:
  - Grupo 1: docentes com formação superior de licenciatura na mesma disciplina que lecionam, ou bacharelado na mesma disciplina com curso de complementação pedagógica concluído.
  - Grupo 2: docentes com formação superior de bacharelado na disciplina correspondente, mas sem licenciatura ou complementação pedagógica.
  - Grupo 3: docentes com licenciatura em área diferente daquela que leciona, ou combacharelado nas disciplinas da base curricular

comum e complementação pedagógica concluída em área diferente daquela que leciona.

- Grupo 4: docentes com outra formação superior não considerada nas localizaçãos anteriores.
- Grupo 5: docentes que não possuem curso superior completo.

- Índice de Esforço Docente: indicador do esforço empreendido pelos docentes da educação básica brasileira no exercício de sua profissão (BRASIL, 2023d). Há 6 níveis possíveis para cada professor:
  - Nível 1: docente que tem até 25 alunos e atua em um único turno, escola e etapa.
  - Nível 2: docente que tem entre 25 e 150 alunos e atua em um único turno, escola e etapa.
  - Nível 3: docente que tem entre 25 e 300 alunos e atua em um ou dois turnos em uma única escola e etapa.
  - Nível 4: docentes que tem entre 50 e 400 alunos e atua em dois turnos, em uma ou duas escolas e em duas etapas.
  - Nível 5: docente que tem mais de 300 alunos e atua nos três turnos, em duas ou três escolas e em duas etapas ou três etapas.
  - Nível 6: Docente que tem mais de 400 alunos e atua nos três turnos, em duas ou três escolas e em duas etapas ou três etapas.

- Percentual de Docentes com Curso Superior
- Média de Alunos por Turma: número total de alunos matriculados em uma escola dividido pelo número total de turmas da instituição.
- Média de Horas-aula Diária: número total de horas-aula oferecidas em uma etapa de ensino dividido pelo número total de dias letivos.
- Taxas de Distorção Idade-série: número de alunos com mais de 2 anos de atraso escolar dividido pelo número total de alunos.
- Taxa de Aprovação: número de alunos aprovados dividido pelo total de alunos matriculados.
- Taxa de Reprovação: número de alunos reprovados dividido pelo total de

alunos matriculados.

- Taxa de Abandono: número de alunos que abandonaram a escola dividido pelo total de alunos matriculados.

2.2.3 Dados Geográficos

Baixados do Github[1], foram extraídos manualmente os GeoJsons de unidade Federativa, mesorregião, microrregião, municípios. Serão utilizados para plotar os gráficos baseadas em Mapa.

2.3 Tecnologias

Aqui apresentamos um compilado de todas as tecnologias que foram utilizadas no desenvolvimento deste trabalho, separando-as entre as três principais áreas que a aplicação abrange.

2.3.1 ETL: Python

Python é uma linguagem de programação de alto nível criada por Guido van Rossum em 1991 (PYTHON.ORG, 2023). Desde então, ela tem se tornado cada vez mais popular (TIOBE.COM, 2023) devido às suas características marcantes e a versatilidade de aplicações em diversos campos.

Entre as principais características do Python, destacam-se a sintaxe clara e concisa, que facilita a leitura e escrita do código, além de tornar a linguagem mais fácil de aprender e usar. A documentação abrangente disponível na internet é outro fator que contribui para a popularidade da linguagem, pois permite aos desenvolvedores encontrar soluções e informações rapidamente. A comunidade ativa e engajada também desempenha um papel importante, promovendo o compartilhamento de conhecimento e a resolução de problemas em conjunto (RAMALHO, 2022).

Outro aspecto positivo do Python é o suporte a programação orientada

[1] Disponível em <https://fititnt.github.io/gis-dataset-brasil/>. Acesso em: 3 abr. 2023.

a objetos, procedural e funcional, oferecendo aos programadores diferentes estilos de programação e permitindo a criação de soluções mais adequadas a cada problema. Essa versatilidade faz com que o Python seja uma excelente escolha para desenvolvedores que trabalham em projetos de diversos tamanhos e complexidades.

Apesar de suas inúmeras qualidades, o Python apresenta algumas deficiências, como a velocidade de execução, que pode ser mais lenta em comparação a outras linguagens, como C e Java. Essa limitação pode ser um problema em aplicações que demandam alto desempenho, como jogos e sistemas embarcados. Além disso, a tipagem dinâmica pode gerar dificuldades no gerenciamento e manutenção de projetos de grande porte. Outra questão relacionada ao Python é a concorrência, uma vez que a linguagem possui um mecanismo chamado Global Interpreter Lock (GIL) que limita a execução simultânea de múltiplas threads, dificultando a implementação de programas paralelos.

Se equilibrando entre as suas qualidades e defeitos, Python tem se destacado em várias áreas, especialmente no desenvolvimento Web e na Ciência de Dados. Na Web, frameworks como Django e Flask facilitam a criação de aplicações robustas e escaláveis. Já na Ciência de Dados, a linguagem tem sido amplamente adotada graças a bibliotecas como NumPy, Pandas, Matplotlib, spaCy e TensorFlow, que fornecem ferramentas poderosas para análise de dados, aprendizado de máquina e inteligência artificial (STANčIN; JOVIć, 2019).

No desenvolvimento deste projeto, Python desempenha um papel central, sendo a base para todas as ferramentas utilizadas no processo de ETL e visualização de dados. A extração dos arquivos CSV do Censo Escolar no portal do INEP foi realizada com o auxílio da biblioteca Requests. Para a limpeza, transformação e carga dos dados em formato Parquet, utilizou-se o Pandas. O DuckDB foi o banco de dados colunar empregado para analisar e consultar os dados salvos em Parquet. Por último, os dados são exibidos no Streamlit, um framework Python dedicado à construção de aplicativos web voltados para dados.

2.3.2 Armazenamento particionado de dados: Apache Parquet

O formato de arquivo Apache Parquet é uma escolha popular no campo da análise de dados devido às suas diversas qualidades e capacidades. Desenvolvido como um projeto de código aberto, esse formato colunar armazena dados em uma estrutura altamente eficiente e compacta, tornando-o particularmente adequado para trabalhar com grandes conjuntos de dados (APACHE.ORG, 2023).

Uma das principais vantagens do Parquet é sua capacidade de compressão, que permite reduzir significativamente o espaço em disco necessário para armazenar informações. Isso é possível graças à sua organização colunar, que agrupa dados semelhantes juntos, facilitando a compressão e melhorando o desempenho na leitura. Outra característica importante do Parquet é sua capacidade de particionar dados. Através do particionamento, os conjuntos de dados são divididos em partes menores, ou partições, com base em colunas específicas. Isso facilita a leitura e o processamento de informações, já que é possível acessar apenas as partições relevantes, reduzindo a quantidade de dados lidos e, consequentemente, otimizando consultas e análises.

O amplo uso do formato Parquet na área de dados pode ser atribuído à sua eficiência e compatibilidade com diversas ferramentas e plataformas de processamento de dados. Tanto soluções de armazenamento de dados em larga escala, como o Hadoop, quanto plataformas de análise de dados, como o Apache Spark e o Dremio, suportam o formato Parquet, o que permite a fácil integração e interoperabilidade entre diferentes sistemas (IVANOV; PERGOLESI, 2020).

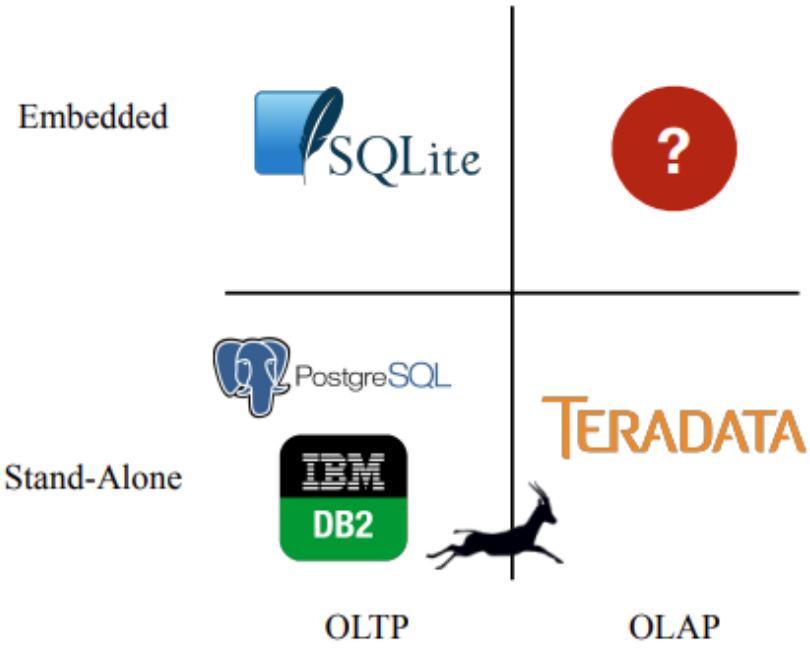
Ao longo da realização deste projeto, a adoção do formato Parquet para armazenamento de dados provou ser a solução mais eficiente. Graças ao seu algoritmo de compressão, os dados transformados puderam ser armazenados diretamente no repositório do projeto no GitHub, facilitando o compartilhamento e a colaboração. Além disso, a visualização quase em tempo real dos gráficos no Streamlit foi viabilizada pelo particionamento de dados por ano, estrutura colunar e rápida descompressão, recursos oferecidos pelo Parquet, permitindo assim uma análise ágil e eficaz dos

dados.

2.3.3 Consulta de dados: DuckDB

DuckDB é um banco de dados relacional para análise colunar, de código aberto desenvolvido em C++ e que visa fornecer uma solução de banco de dados que seja rápida, fácil de usar e que possa ser executada em várias plataformas, incluindo sistemas operacionais Linux, Windows e macOS (DUCKDB.ORG, 2023). Surgiu principalmente devido a ausência de um banco de dados OLAP para sistemas embedados (RAASVELDT; MüHLEISEN, 2019).

Figura 2.4: Ausência de um SGBD OLAP para sistemas embarcados.



Fonte: Raasveldt e Mühleisen (2019, p. 1).

Algumas das principais características do DuckDB incluem o suporte a consultas SQL completas, um mecanismo de armazenamento de coluna otimizado para consultas analíticas e científicas de dados, capacidade de execução de consultas distribuídas, compatibilidade com JDBC e ODBC e integração com Python através de uma biblioteca Python nativa.

Uma das principais vantagens do DuckDB é seu desempenho. Ele é projetado para ser rápido, e tem sido comprovado que supera muitos bancos

de dados relacionais populares em termos de velocidade de execução de consultas analíticas e científicas de dados. Além disso, o DuckDB é muito leve em termos de recursos de hardware necessários, tornando-o ideal para ambientes onde recursos são limitados. Outra vantagem do DuckDB é sua facilidade de uso. Ele possui uma interface simples e fácil de usar para a execução de consultas SQL, e a integração com outras linguagens de programação, como Python, é simples e direta.

Por fim, o DuckDB é um banco de dados relativamente novo, e sua comunidade ainda está crescendo. No entanto, ele já possui uma base sólida de usuários e desenvolvedores e tem sido amplamente utilizado em aplicações analíticas e científicas de dados, bem como em aplicativos de negócios. É uma opção interessante para aqueles que buscam um banco de dados leve, rápido, fácil de usar e com recursos de análise de dados avançados. Sua arquitetura colunar, suporte a consultas distribuídas e integração com Python o tornam uma opção promissora para muitos casos de uso.

No contexto deste projeto, a habilidade de executar consultas SQL em arquivos Parquet e a funcionalidade de análise colunar se mostraram extremamente úteis, permitindo uma experiência fluida e ágil na utilização da aplicação de dados construída em Streamlit.

2.3.4 Aplicação de dados: Streamlit

Streamlit é uma biblioteca open-source em Python que permite criar aplicações web interativas com facilidade e rapidez. Desenvolvida para ajudar cientistas de dados e desenvolvedores a compartilhar insights e visualizações, a plataforma é conhecida por sua simplicidade e elegância (KHORASANI et al., 2022).

Uma das principais vantagens do Streamlit é a capacidade de criar interfaces de usuário ricas e interativas sem a necessidade de conhecimento aprofundado em front-end ou linguagens como HTML, CSS e JavaScript. A biblioteca se baseia em um conjunto de componentes prontos, como sliders, botões e gráficos, que podem ser facilmente incorporados ao código (STREAMLIT.IO, 2023).

O Streamlit é executado localmente no computador do usuário, o que facilita o desenvolvimento e teste de aplicações antes de implantá-las em servidores ou na nuvem. Além disso, a plataforma oferece integração nativa com diversas bibliotecas de análise e visualização de dados, como Pandas, NumPy, Matplotlib e Plotly.

A arquitetura do Streamlit é orientada a eventos, o que significa que as aplicações são atualizadas automaticamente sempre que ocorre uma interação do usuário. Isso permite que os desenvolvedores criem aplicações dinâmicas e responsivas com poucas linhas de código.

No entanto, apesar de suas vantagens, o Streamlit também possui algumas limitações. A principal delas é a falta de suporte para aplicações multiusuário em tempo real, já que cada interação é tratada como uma nova sessão. Além disso, a biblioteca não oferece tantos componentes de interface do usuário personalizados quanto outras soluções mais robustas, como Dash ou Shiny.

Em resumo, o Streamlit é uma solução elegante e fácil de usar para criar aplicações web interativas em Python. Sua simplicidade e conjunto de recursos o tornam uma opção atraente para cientistas de dados e desenvolvedores que desejam compartilhar análises e visualizações sem se aprofundar no desenvolvimento de front-end. Entretanto, para aplicações mais complexas e com múltiplos usuários, outras soluções podem ser mais adequadas.

Neste trabalho, Streamlit possibilitou criar de forma low-code uma plataforma interativa de apresentação dos dados do Censo Escolar. Sua integração com DuckDB garantiu um maneira veloz de analisar os dados.

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados cinco trabalhos cujos materiais de estudo ou objetivos possuem semelhanças com os deste trabalho.

3.1 Determinantes da repetência escolar no Brasil

Nesse texto do Instituto de Pesquisa Econômica Aplicada (Ipea), Oliveira e Soares (2012) apresentam os principais determinantes da repetência no Ensino Fundamental usando os Censos Escolares entre 2007 e 2010. Inicialmente os autores analisam a perda de observações sobre um aluno ao longo do período observado, fato que eles chamaram de atrito. Usando um modelo Logit para verificar a correlação entre as variáveis do Censo de um ano (sexo, defasagem idade-série, usuário de transporte, etc) e probabilidade de atrito no ano seguinte, os pesquisadores não encontraram correlação de atrito com nenhuma das variáveis, concluindo que esse fenômeno ocorre de forma aleatória e, por consequência, não compromete a qualidade dos dados na busca do objetivo final, que é explicar a repetência de um ano em função do Censo Escolar do ano anterior.

Na busca dessa resposta, foi utilizado novamente o modelo Logit. As variáveis explicativas se dividem entre características de alunos, turmas, professores, escola, estado federativo e ano. Devido a enorme quantidade de variáveis sobre a infraestrutura da escola, foi utilizado Análise da Componente Principal (PCA) para resumir esse conjunto. Além disso, os dados foram separados entre alunos dos anos iniciais do Ensino Fundamental (1º ao 5º ano) e os alunos dos anos finais (6º ao 9º ano), buscando também se investigar a influência dos ciclos educacionais na probabilidade de repetência.

Ao analisar o modelo, constatou-se que os alunos do sexo masculino apresentam maior propensão à repetência em comparação às alunas, e que os estudantes com necessidades especiais (PNEs) ou que dependem do transporte escolar público têm maior probabilidade de fracassar. Em relação às escolas, aquelas localizadas em São Paulo e com infraestrutura adequada apresentam menor incidência de repetência. Além disso, os estudantes

matriculados em escolas administradas pelos municípios estão mais propensos a ter dificuldades escolares. Concluiu-se também que uma política de aumento do número de horas-aula seria mais eficaz do que a redução do número de alunos por turma, como já comprovado em estudos anteriores.

3.2 O impacto do programa bolsa família sobre a repetência

Os autores deste estudo, Oliveira e Soares (2013), avaliam como o Programa Bolsa Família (PBF) afeta a repetição escolar no Ensino Fundamental. Eles usaram informações do Cadastro Único (CadÚnico), Projeto Frequência e Censo Escolar do ano 2008.

Na análise das bases, foi verificado que não há uma chave de identificação única de tal maneira que se localize, com perfeição, o mesmo aluno nas três bases. A melhor intersecção encontrada foi a variável de identificação da escola 3.1.

Figura 3.1: Sobreposição de informações entre as bases e possível chave em comum

Fonte: Oliveira e Soares (2013, p. 9)

Devido as limitações dos dados, os autores usaram três abordagens

diferentes. A primeira é estimar essa probabilidade por meio de regressão logística, usando apenas o universo do CadÚnico. A segunda forma é usar análise de regressão para estimar o impacto do valor do benefício na repetência. A terceira é medir o impacto do percentual de crianças beneficiárias sobre a repetência usando escolas como unidade de análise.

Analisando os resultados das 3 abordagens, foi constatado que alunos participantes do programa possui uma probabilidade de repetência 11% menor do que os demais. Em relação aos demais fatores, a repetência foi maior entre alunos PNEs, meninos, naqueles que estão em famílias menos escolarizadas, nas domiíclio menos favorecidos, nas escolas menos estruturadas ou que dependem administrativamente dos municípios e estados.

3.3 Que indicadores influenciam na qualidade da educação da Paraíba?

O estudo dos autores Silva et al. (2022) busca demonstrar as limitações que se tem ao utilizar somente o Índice de Desenvolvimento da Educação Básica (Ideb) para medir a qualidade da educação básica no Brasil, comparando-o com os demais indicadores provenientes do Censo Escolar. Para isso foram utilizados os dados de 300 escolas do Ensino Médio da Paraíba. Usando a linguagem de programação R, foram construídos modelos lineares generalizados (GLM) para verificar a relação do Ideb com as variáveis: percentual de professores no grupo 1 de adequação da formação docente, média de alunos por turma, nível de complexidade de gestão escolar, média de horas-aula diária, percentual de de professores no nível de esforço docente, docentes com curso superior, regularidade docente e taxa de distorção idade-série. O fator de inflação (VIF) foi usado para determinar a colinearidade entre as variáveis independentes, removendo as que possuíam VIF menor que 4. Para a escolha do melhor modelo, foi comparado os valores de Critério de Informação de Akaike (AIC) dos modelos descritos acima e outros modelos nulos (sem variáveis preditoras), considerando apenas aqueles com valores de AIC maior que 655.93.

Os resultados indicaram que a média de horas aula diárias tem um efeito positivo na qualidade da educação da Paraíba, onde cada unidade de

horas de aula aumenta cerca de três vezes a probabilidade do Ideb também aumentar. A variável complexidade de gestão escolar, que representa o porte e a quantidade de turnos das escolas, apresentou efeito positivo na qualidade da educação em todos os níveis de gestão, exceto no nível cinco, que são escolas de porte entre 150 e 1000 matrículas, operando em três turnos, com duas ou três etapas, apresentando a EJA como etapa mais elevada. A taxa de distorção idade-série demonstrou efeito negativo na qualidade da educação, quanto maior a taxa de distorção menor é a nota do Ideb das escolas investigadas. O modelo apresentou odds-ratio (razão de probabilidade) de -9.079, isto é, cada unidade de distorção, diminui cerca de nove vezes a probabilidade do Ideb das escolas aumentarem.

3.4 Modelagem e Visualização Científica de Dados Educacionais

A pesquisa dos autores Barros et al. (2017) investiga técnicas de visualização de dados em conjunto com técnicas de estatísticas e de aprendizagem de máquina, com o objetivo final de potencializar a interpretação de dados educacionais. Para essa tarefa foram utilizados os dados educacionais de alunos dos cursos de Licenciatura em Espanhol e Técnico de Segurança do Trabalho do Instituto Federal do Rio Grande do Norte no período de 2008 a 2016.

Iniciando com a utilização dos gráficos boxplot, violino e Q-Q, teve-se uma visão geral da distribuição das médias finais da disciplina. Posteriormente, empregou-se o algoritmo k-means para clusterizar os dados e, uma vez atribuídas as classes, realizou-se uma nova análise por meio do gráfico Q-Q e do Mapa Perceptual gerado pela Análise de Correspondência.

Nos resultados finais ficou constatado a limitação do gráfico boxplot em apresentar uma distribuição bimodal das médias, fenômeno este que foi corretamente visualizado no gráfico violino. Uma vez que foi detectado a possível presença de 2 grupos, K-means conseguiu com sucesso classificar os dados entre esses grupos. O gráfico Q-Q evidencia que a normalidade da distribuição não é viável na distribuição não clusterizada, contudo, na distribuição clusterizada, tal possibilidade se concretiza. Utilizando a técnica de análise de correspondência foi demonstrado que os alunos que possuem o

label "Aprovado"na base de dados original são os alunos classificado pelo k-means como classe 1, e os alunos com label "Reprovado"são os classificados pela classe 0.

3.5 Dados Abertos Educacionais Brasileiros: Um Mapeamento Sistemático da Literatura

O trabalho de Ferreira et al. (2021) tem como objetivo realizar um mapeamento sistemático de artigos científicos que abordam dados abertos educacionais do Brasil, produzidos entre 2010 e julho de 2021.

Utilizando strings de buscas como “mineração de dados”, “censo escolar”, “enade, “saeb”, etc, junto com suas respectivas traduções em Inglês, foram retirados estudos das bases IEEE, ScienceDirect, ACM Digital, SpringerLink, Scopus, Periódicos da Capes e Google Acadêmico. Foram removidos aqueles que não são sobre a educação brasileira, que não estão em Português ou Inglês, trabalhos de conclusão de graduação e pós-graduação, que não estão acessíveis integralmente, relatórios de workshop, relatórios técnicos, etc. Por fim, foram lidos o título, resumo ou abstract e palavras-chave, dos quais se selecionaram 210 candidatos, e após a leitura completa dos artigos, 91 deles foram classificados.

Os resultados finais indicaram que o ano de 2019 registrou o maior número de publicações, seguido por 2020 e 2018. A base de dados mais frequente foi a do Enem, seguida pelo Enade, Prova Brasil e Censo da Educação Básica. As ferramentas mais empregadas foram, principalmente, Weka e R, seguidas de SPSS e Python. Os algoritmos de maior destaque foram, sobretudo, J48 e Naive Bayes (Figura 3.2. Quanto às metodologias de mineração de dados, o KDD obteve 54% das pesquisas classificadas e o CRISP-DM, 46% dos estudos.

3.6 Análise Comparativa

Os 2 primeiros estudos são parecidos entre si compartilhando a autoria de OLIVEIRA; SOARES, e objetivo, que é prever a repetência

Figura 3.2: Nuvem de palavras dos algoritmos mais citados em artigos sobre mineração de dados educacionais brasileiros

Fonte: Ferreira et al. (2021, p. 1192)

utilizando somente ou não somente o Censo Escolar. No primeiro foi marcado exatamente as variáveis provenientes do Censo que são utilizadas no modelo de previsão, enquanto que no segundo não. Neste presente trabalho também foram utilizados dados sobre escolas. Os dados granularizados a nível de aluno não foram utilizados neste presente artigo, pois não estão mais disponíveis, pelo motivo do INEP considerar que elas vão contra a LGPD, conforme discutido na subseção 2.1.5.

No terceiro estudo (SILVA et al., 2022), utilizando o Censo Escolar, é problematizada a não utilização de outros indicadores educacionais além do Ideb, destacando-se a média de horas-aula e taxa de distorção idade-série, indicadores estes que serão abordados neste presente artigo.

O quarto trabalho (BARROS et al., 2017), embora aborde uma base de dados dados que não é Censo, ainda assim consegue demonstrar o quão útil a análise de dados pode ser na tomada de decisões sobre a educação. Além disso, destacou a importância de definir corretamente um gráfico numa apresentasão de dados.

Finalizando, no artigo de FERREIRA et al. foi ilustrado que a mineração de dados educacionais é uma área de pesquisa em ascensão e necessária no

desenvolvimento de processos educacionais.

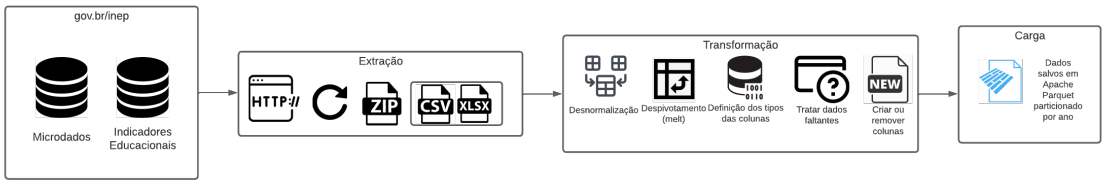
4 DESENVOLVIMENTO

Este Capítulo detalha a implementação da extração, transformação/carga e visualização dos dados do Censo Escolar. Na Seção 4.1 há um descrição geral, com a etapa de Extração sendo abordada em detalhes na Seção 4.2, e a tapa de Transformação na Seção 4.3. Na Seção 4.4 é apresentado a etapa de Visualização.

4.1 Visão Geral

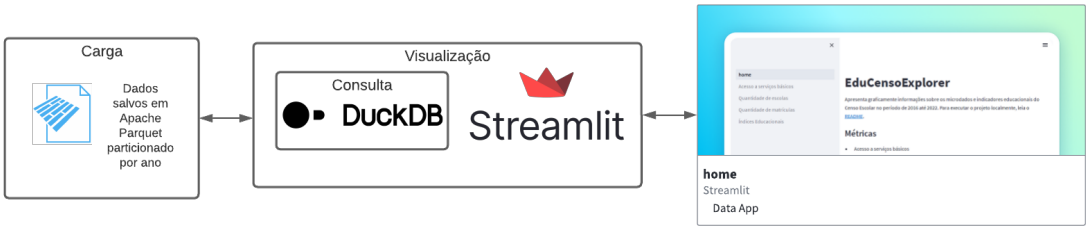
O fluxo dos dados utilizados por este projeto inicia no portal do INEP[1], onde estão hosperados os microdados e indicadores educacionais do Censo Escolar, e que são extraídos e transformados via Python. Após isso, os dados são armazenados no formato de arquivo Parquet para então, por meio do SGBD DuckDB, serem consultados pela aplicação Web criada com Streamlit.

Figura 4.1: Visão alto-nível do trabalho desenvolvido - parte inicial



Fonte: o autor

Figura 4.2: Visão alto-nível do trabalho desenvolvido - parte final



Fonte: o autor

Este trabalho foi desenvolvido em um computador que possui as seguintes especificações: sistema operacional PopOs 22.04 LTS, processador Intel de 12ª geração e 16 GB de RAM. Para assegurar o isolamento do

[1] Disponível em <https://www.gov.br/inep/pt-br/areas-de-atuacao/pesquisas-estatisticas-e-indicadores/censo-escolar>. Acesso em 07 abr. 2023

ambiente de desenvolvimento foi considerado inicialmente Docker, porém como todas as ferramentas utilizadas no trabalho são baseadas em Python, *Virtualenv*, um pacote simples do Python, já demonstrou ser o suficiente para isolar o ambiente. O processo de configuração pode ser lido no *README* do projeto.

O processo de ETL foi escrito para processar os dados de forma total, não incremental, dado a alta granularidade de atualização dos dados, que é 1 ano. O código fonte está disponível no repositório deste projeto[2].

4.2 Extração

Nessa seção é explicado com foi implementada a extração dos dados utilizados pelo trablho.

4.2.1 Microdados

No portal do INEP, os microdados estão em uma URL que varia conforme o ano, mas que possui o mesmo padrão, cujo prefixo é **download.inep.gov.br/dados_abertos/microdados_censo_escolar** e o sufixo é **_<YEAR>.zip**, com <YEAR> sendo o *placeholder* para a variável ano. Com a URL formada é possível fazer a requisição HTTP, que neste trabalho foi feita através da biblioteca ***requests***. Além disso, por se tratar de uma API, é possível que uma requisição a ela falhe em algum momento. Para lidar de forma elegante com essas falhas, foi utilizado o decorador ***on_exception*** da biblioteca ***backoff***. Esse decorador permite definir retentativas de função em caso de falhas, tornando o processo de lidar com as falhas da API mais eficiente.

O arquivo que será baixado será do tipo ZIP. Umas das formas de verificar se um arquivo ZIP foi baixado corretamente é simplemeste abrindo ele, já que é retornado uma *exception* em caso de um arquivo corrompido (código 4.1, linha 16).

A função ***download_file*** (código 4.1, linha 20) faz a coordenação entre

[2]Disponível em <https://github.com/rjribeiro/EduCensoExplorer>

make_request (código 4.1, linha 7) e ***test_zip*** (código 4.1, linha 16). Em caso de falha, é aguardado 100 segundos, para então esse processo ser tentado mais uma única vez. Se ele falhar além disso, então é necessário a intervenção humana para entender o motivo da falha.

Em seguida, a função ***unzip_file*** (código 4.1, linha 33) extrai todos os arquivos dentro do ZIP baixado. Cada ano terá um CSV com nome diferente contendo os microdados. Como esses nomes não seguem um padrão bem definido, e para facilitar a etapa de transformação, o CSV é renomeado para o seu respectivo ano e movido para o diretório **data/raw/microdados**.

Listing 4.1: Extração dos microdados. Disponível em <https://github.com/rjribeiro/EduCensoExplorer/blob/main/etl/microdados/extract.py>

```

@backoff.on_exception(
    backoff.expo,
    requests.exceptions.RequestException,
    max_tries=2
)
def make_request(year: int) -> None:
    url = f"https://download.inep.gov.br/dados_abertos/
    microdados_censo_escolar_{year}.zip"
    os.makedirs("./data/raw/microdados/zips", exist_ok=True)
    with requests.get(url, stream=True, verify=False, timeout=600) as r:
        r.raise_for_status()
        with open(f"./data/raw/microdados/zips/{year}.zip", 'wb') as f:
            for chunk in r.iter_content(chunk_size=8192):
                f.write(chunk)

def test_zip(year: int) -> None:
    ZipFile(f"./data/raw/microdados/zips/{year}.zip", 'r')


def download_file(year: int) -> None:
    url = f"https://download.inep.gov.br/dados_abertos/
    microdados_censo_escolar_{year}.zip"
    logger.info(f"Downloading {url}")
    try:
        make_request(url)
        test_zip(year)
    except (requests.exceptions.ChunkedEncodingError, BadZipfile):
        sleep(100)
        if f"./data/raw/microdados/zips/{year}.zip" in os.listdir():
            os.remove(f"{year}.zip")
        make_request(url)
        test_zip(year)

def unzip_file(year: int) -> None:
    logger.debug("Unzipping")
    with ZipFile(f"./data/raw/microdados/zips/{year}.zip", 'r') as zip:
        zip.extractall("data/raw/microdados")
        csv_file = [file for file in zip.namelist() if ".csv" in file.lower()]
```

```
38          zip.extractall("data/raw/microdados", members=csv_file)
39          os.rename(
40              f"./data/raw/microdados/{csv_file[0]}",
41              f"./data/raw/microdados/{year}.csv"
42          )
```

4.2.2 Indicadores Educacionais

Seguindo a definição de qualidade de Dados Abertos de Tim Berners-Lee (já apresentada em 2.3), as bases de Indicadores Educacionais da educação brasileira conseguiriam somente 1 estrela, numa escala de 5, porque os dados estão em planilhas do tipo Excel que, além ser um formato proprietário, dificulta bastante a transformação dos dados. Além disso, na base **Taxa de Rendimento Escolar**, que neste trabalho chamaremos de **TRE**, não há um padrão bem definido para a URL de download, com o formato variando conforme o ano. A função ***get_url*** (código 4.2, linha 2) abstrai a definição da URL, conforme indicador e ano. O restante do script possui a mesma estrutura geral que o script usado na extração dos microdados (4.2.1). Os indicadores extraídos foram os mesmos descritos na Seção 2.2.2. Cada combinação de indicador e ano possui 2 planilhas Excel: uma contendo informações agregadas a nível municipal e outra agregada a nível estadual, regional e federal. O indicador **TRE** para o ano 2022 ainda não estava disponível até a data da escrita deste trabalho. No ano de 2016 não há informações a nível estadual e regional.

Listing 4.2: Extração dos indicadores educacionais. Disponível em <https://github.com/rjribeiro/EduCensoExplorer/blob/main/etl/indicadores/extract.py>

```
1
2 def get_url(indicador: str, base: str, year: int) -> str:
3     if indicador in {"AFD"}:
4         base = base.replace("UFS", "UF")
5     if indicador == "TRE":
6         if year in {2016, 2017}:
7             url = f"https://download.inep.gov.br/informacoes_estatisticas/
    indicadores_educacionais/{year}/TAXA_REND_{year}_{base.upper()}.zip"
```

```
        elif year == 2018:
            url = f"https://download.inep.gov.br/informacoes_estatisticas/indicadores_educacionais/2018/TX_REND_{base.upper()}_2018.zip"
        elif year in range(2019, 2022):
            url = f"https://download.inep.gov.br/informacoes_estatisticas/indicadores_educacionais/{year}/tx_rend_{base.lower()}_{year}.zip"
        else:  # TRE 2022 ainda nao disponivel
            raise requests.exceptions.HTTPError(f"{indicador}/{year}")
    else:
        url = f"https://download.inep.gov.br/informacoes_estatisticas/indicadores_educacionais/{year}/{indicador}_{year}_{base.upper()}.zip"
    return url


@backoff.on_exception(
    backoff.expo,
    requests.exceptions.RequestException,
    max_tries=2
)
def make_request(year: int, indicador: str, base: str) -> None:
    url = get_url(indicador, base, year)
    os.makedirs(f"./data/raw/{indicador}/{base}/zips", exist_ok=True)
    with requests.get(url, stream=True, verify=False, timeout=600) as r:
        r.raise_for_status()
        with open(f"./data/raw/{indicador}/{base}/zips/{year}.zip", "wb") as f:
            for chunk in r.iter_content(chunk_size=8192):
                f.write(chunk)


def test_zip(year: int, indicador: str, base: str) -> None:
    ZipFile(f"./data/raw/{indicador}/{base}/zips/{year}.zip", "r")


def download_file(year: int, indicador: str, base: str) -> None:
    logger.info(f"Downloading {indicador}/{base}/{year}")
    try:
        make_request(year, indicador, base)
        test_zip(year, indicador, base)

    except (requests.exceptions.ChunkedEncodingError, BadZipfile):
        sleep(100)
```

```
46          if f"./data/raw/{indicador}/{base}/zips/{year}.zip" in os.listdir():
47              os.remove(f"./data/raw/{indicador}/{base}/zips/{year}.zip")
48          make_request(year, indicador, base)
49          test_zip(year, indicador, base)
50
51
52  def unzip_file(year: int, indicador: str, base: str) -> None:
53      logger.debug("Unzipping")
54      folder = f"data/raw/{indicador}/{base}/{year}"
55      with ZipFile(f"./data/raw/{indicador}/{base}/zips/{year}.zip", "r") as zip:
56          excel_file = [file for file in zip.namelist() if "xlsx" in file.lower()
        ]
57          zip.extractall(folder, members=excel_file)
58          os.rename(
59              f"{folder}/{excel_file[0]}",
60              f"{folder}.xlsx"
61          )
62          rmtree(folder)
```

4.3 Transformação e Carga

Nessa seção é explicado com foi implementada a transformação e carga dos dados utilizados pelo trabalho.

4.3.1 Microdados

A principal tarefa nessa etapa é adequar os dados ao tipo correto, conforme explicado na tabela 2.1. O Pandas possui uma inferência de *schema*, mas que não funciona corretamente na maioria das vezes. Nos microdados a maioria dos colunas é interpretada automaticamente com *String*.

Colunas que informam sobre quantidade, que são aquelas com prefixo **QT**, são convertidas para o tipo *Int32*. Dados faltantes são preenchidos com o valor 0 (código 4.3, linha 1).

Para as colunas com prefixo **NU** o tratamento varia. A coluna **NU_ANO_CENSO** foi convertida para *Int16*, enquanto que as demais

continuaram no tipo *String* (código 4.3, linha 1).

Colunas identificadoras são aquelas que possuem prefixo **CO**. As utilizadas posteriormente como chaves para *joins* foram convertidas para *Int32* com o objetivo de tornar as junções mais rápidas. As demais continuaram como *String* (código 4.3, linha 13).

Colunas com prefixo **TP**, que aqui chamarei de categóricas nominais, são mapeadas de valores numéricos discretos para seus respectivos valores dentro dentro da sua categoria nominal. O mapa foi elaborado a partir do dicionário de dados dos microdados de 2022. O Jupyter Notebook que constrói o mapa pode ser conferido no Github do projeto[3]. Será lançado uma exceção durante a execução do script caso seja encontrado alguma que não tenha sido mapeada previamente. Colunas sobre línguas indígenas, prefixo **CO_LINGUA_INDIGENA**, também foram consideradas como categóricas nominais (código 4.3, linha 24).

Colunas que possuem data são convertidas para o tipo *Datetime* usando **"%d%b%Y:%H:%M:%S"** como *string pattern*, ou para *Null* caso isso não seja possível (código 4.3, linha 46).

Colunas booleans, prefixo IN, que possuiam 0 ou 1, foram convertidas para o tipo *Bool* (código 4.3, linha 46).

Para facilitar a visualização dos dados a nível federal, é criada uma nova coluna chamada **NO_PAIS** que possui somente a string "Brasil"(código 4.3, linha 69).

Finalizado a transformação, resta a etapa de carga, que neste projeto se resume a salvar os dataframes transformados no formato Apache Parquet. A coluna que contém o ano do censo (**NU_ANO_CENSO**) foi a única utilizada para o particionamento, pelo motivo que somente ela garante um particionamento balanceado, sem resultar em arquivos muito pequenos que poderiam comprometer a agilidade de consulta na etapa de visualização (código 4.3, linha 74).

Listing 4.3: Transformação dos microdadados. Disponível em <https://github.com/rjribeiro/EduCensoExplorer/blob/main/etl/microdados/transform.py>

```
1 def transform_integer_columns(df: pd.DataFrame) -> pd.DataFrame:
2     integer_columns = [column for column in df.columns
```

[3]Disponível em <https://github.com/rjribeiro/EduCensoExplorer/blob/main/etl/microdados/transform/map_categorial_columns.ipynb>

```
                        if column.startswith("QT")]
    df[integer_columns] = df[integer_columns]. \
        fillna(0). \
        astype("int32")

    df["NU_ANO_CENSO"] = df["NU_ANO_CENSO"]. \
        astype("int16")

    return df

def transform_identifier_columns(df: pd.DataFrame) -> pd.DataFrame:
    identifier_columns = ["NU_DDD", "NU_TELEFONE", "NU_CNPJ_ESCOLA_PRIVADA",
                          "NU_CNPJ_MANTENEDORA", "CO_ESCOLA_SEDE_VINCULADA", "
    CO_IES_OFERTANTE",
                          "CO_DISTRITO", "CO_CEP", "CO_ENTIDADE"]
    df[identifier_columns] = df[identifier_columns].astype("str")

    identifier_columns_int = ["CO_REGIAO", "CO_UF", "CO_MESORREGIAO", "
    CO_MICRORREGIAO", "CO_MUNICIPIO"]
    df[identifier_columns_int] = df[identifier_columns_int].astype("int32")

    return df

def transform_categorical_columns(df: pd.DataFrame) -> pd.DataFrame:
    categorical_columns = [column for column in df.columns if column.startswith
    ("TP")] + [
        "CO_LINGUA_INDIGENA_1", "CO_LINGUA_INDIGENA_2", "CO_LINGUA_INDIGENA_3"
    ]
    with open("./etl/microdados/transform/map_categorical_columns.json") as
    file:
        map_categorical_columns = json.load(file)

    if columns_not_mapped := set(categorical_columns).difference(
            map_categorical_columns.keys()):
        raise Exception(f"Columns not mapped: {columns_not_mapped}")
    df[categorical_columns] = df[categorical_columns]. \
        astype("float"). \
        fillna(-1). \
        astype("int"). \
        astype("str")
    for column in categorical_columns:
```

```
        df[column] = df[column]. \
            replace(to_replace=map_categorical_columns[column])
    df[categorical_columns] = df[categorical_columns].replace(["-1", "9"], None
    )
    return df

def transform_date_columns(df: pd.DataFrame) -> pd.DataFrame:

    date_columns = [column for column in df.columns if column.startswith("DT")]
    for column in date_columns:
        df[column] = df[column]. \
            apply(
            lambda date:
            datetime.strptime(date, "%d%b%Y:%H:%M:%S")
            if isinstance(date, str) and date != "0"
            else None
        )
    return df


def transform_boolean_columns(df: pd.DataFrame) -> pd.DataFrame:

    boolean_columns = [column for column in df.columns
                       if column.startswith("IN")]
    df[boolean_columns] = df[boolean_columns]. \
        replace(to_replace=9.0, value=None). \
        astype("bool")
    return df


def create_new_columns(df: pd.DataFrame) -> pd.DataFrame:
    df["NO_PAIS"] = "Brasil"
    return df


def save_dataframe(df: pd.DataFrame) -> None:
    df.to_parquet(
        "./data/transformed/microdados.parquet",
        engine="pyarrow",
        compression="snappy",
        index=None,
```

```
80          partition_cols=["NU_ANO_CENSO"]
81      )
```

4.3.2 Indicadores Educacionais

A leitura é feita conforme o indicador, ano e base (município ou federal/regional/estadual). É necessário cortar o início e o fim das tabelas, com a quantidade de linhas para cortar no início variando conforme o indicador (código 4.4, linha 8). O indicador **TRE** se divide em 3 indicadores dentro de sua planilha: **Taxa de Aprovação** (**TAP**), **Taxa de Reprovação TRP**) e **Taxa de Abandono** (**TAB**), com cada segmento contendo 18 colunas (código 4.4, linha 18).

Iniciando a transformação está a criação, exclusão e renomeação dos colunas. Se a base for municipal, são deletadas as colunas **Região**, **Sigla** e **Código do Município** (código 4.4, linha 58). As colunas restantes se dividem entre 2 tipos, onde as 4 primeiras colunas serão as responsáveis por identificar unicamente cada linha (ano do censo, nome da localidade geográfica, tipo da categoria de escola e tipo de dependência administrativa), e são as métricas do indicador, se dividindo em grupos com tamanho variando conforme o indicador. Embora o significado e a ordem das colunas identificadoras seja sempre o mesmo, os nomes variam de acordo com o indicador e ano. Por causa disso os nomes foram padronizados para **NU_ANO_CENSO**, **NO_LOCALIDADE_GEOGRAFICA**, **NO_CATEGORIA**, e **NO_DEPENDENCIA** (código 4.4, linha 63). Em relação aos nomes das colunas de métricas, a variação é mais intensa e difícil de mapear, mas a ordem também é sempre igual. No Github do projeto[4] pode ser visualizado o mapeamento para cada indicador (código 4.4, linha 70). Como nova coluna é adicionado **TP_LOCALIDADE_GEOGRAFICA**, que classifica a localidade geográfica em município, unidade federativa, região geográfica ou país (código 4.4, linha 76).

Para deixar o dataframe de todos os indicadores com o mesmo número de colunas é realizada uma etapa de despivotamento. As colunas de métricas,

[4] Disponível em <https://github.com/rjribeiro/EduCensoExplorer/blob/main/etl/indicadores/map_indicadores.json>

que variam em quantidade para cada indicador, são reduzidas a 2 colunas: uma contendo o nome da coluna de métrica e outra contendo o seu respectivo valor. Além de padronizar o *schema*, essa transformação ajuda bastante durante a visualização dos dados (código 4.4, linha 89). Por fim os dataframes das bases município e federal/regional/estadual são unidos em uma única tabela (código 4.4, linha 45).

A tabela final então é salva em formato Parquet, particionando pela coluna **NU_ANO_CENSO** (código 4.4, linha 103).

Listing 4.4: Transformação dos indicadores. Disponível em <https://github.com/rjribeiro/EduCensoExplorer/blob/main/etl/indicadores/transform.py>

```
1
2  INDICADORES = {...}
3
4  with open("./etl/indicadores/map_indicadores.json") as f:
5      MAP_INDICADORES = json.load(f)
6
7
8  def load_dataframe(indicador: str, year: int, base: str) -> pd.DataFrame:
9      if indicador in {"TAP", "TRP", "TAB"}:
10         file = f"./data/raw/TRE/{base}/{year}.xlsx"
11     else:
12         file = f"./data/raw/{indicador}/{base}/{year}.xlsx"
13     if not os.path.isfile(file):
14         raise FileNotFoundError(file)
15     match indicador:
16         case "ATU" | "HAD" | "TDI":
17             df = pd.read_excel(file, skiprows=8, skipfooter=6, na_values=["--"
       ])
18         case "TAP" | "TRP" | "TAB":
19             df = pd.read_excel(file, skiprows=8, skipfooter=6, na_values=["--"
       ])
20             offset = 7 if base == "MUNICIPIOS" else 4
21             match indicador:
22                 case "TAP":
23                     df = pd.concat([df.iloc[:, :offset], df.iloc[:, offset
       +(18*0):offset+(18*1)]], axis=1)
24                 case "TRP":
25                     df = pd.concat([df.iloc[:, :offset], df.iloc[:, offset
       +(18*1):offset+(18*2)]], axis=1)
```

```
26                 case "TAB":
27                     df = pd.concat([df.iloc[:, :offset], df.iloc[:, offset
    +(18*2):offset+(18*3)]], axis=1)
28 
29         case "DSU":
30             df = pd.read_excel(file, skiprows=9, skipfooter=6, na_values=["--"
    ])
31         case _:
32             df = pd.read_excel(file, skiprows=10, skipfooter=6, na_values=["--"
    ])
33     return df
34 
35 
36 def transform_dataframe(df_municipios: pd.DataFrame, df_brasil: pd.DataFrame,
    indicador: str) -> pd.DataFrame:
37     df_municipios = get_renamed_and_news_columns(df_municipios, indicador, "
    MUNICIPIOS")
38     df_municipios = get_melted_dataframe(df_municipios)
39     df_municipios = get_dataframe_with_forced_schema(df_municipios)
40 
41     df_brasil = get_renamed_and_news_columns(df_brasil, indicador, "
    BRASIL_REGIOES_UFS")
42     df_brasil = get_melted_dataframe(df_brasil)
43     df_brasil = get_dataframe_with_forced_schema(df_brasil)
44 
45     return pd.concat([df_municipios, df_brasil], axis=0)
46 
47 
48 def get_tipo_localidade_geografica(localidade_geografica: str) -> str:
49     if localidade_geografica.lower() == "brasil":
50         return "Pa s"
51     elif localidade_geografica.lower() in {"norte", "centro-oeste", "nordeste",
     "sul", "sudeste"}:
52         return "Regi o Geogr fica"
53     else:
54         return "Unidade Federativa"
55 
56 
57 def get_renamed_and_news_columns(df: pd.DataFrame, indicador: str, base: str)
    -> pd.DataFrame:
58     # drop
```

```
    if base == "MUNICIPIOS":
        df = df.drop(df.columns[[1, 2, 3]], axis=1)
    df.columns = map(str, range(df.columns.size))


    # renamed columns
    columns = ['NU_ANO_CENSO', 'NO_LOCALIDADE_GEOGRAFICA', 'NO_CATEGORIA', '
    NO_DEPENDENCIA']
    df = df.rename(
        columns=dict(zip(df.columns[:4], columns))
    )


    if len(MAP_INDICADORES[indicador]) == len(df.columns) - 4:
        raise Exception("Quantidade de grupos previamente mapeados n o confere
     com a quantidade de grupos no df")
    df = df.rename(
        columns=dict(zip(df.columns[4:], MAP_INDICADORES[indicador]))
    )
    # new columns
    if base == "MUNICIPIOS":
        df.insert(2, "TP_LOCALIDADE_GEOGRAFICA", "Munic pio")
    else:
        df.insert(
            2,
            "TP_LOCALIDADE_GEOGRAFICA",
            df["NO_LOCALIDADE_GEOGRAFICA"].apply(get_tipo_localidade_geografica
    )
        )

    return df


def get_melted_dataframe(df: pd.DataFrame) -> pd.DataFrame:
    id_vars = df.columns[:5]
    value_vars = df.columns[5:]
    df = df.melt(id_vars=id_vars, value_vars=value_vars, var_name="TP_GRUPO",
    value_name="METRICA")
    return df


```

```
96 def get_dataframe_with_forced_schema(df: pd.DataFrame) -> pd.DataFrame:
97     integer_columns = ["NU_ANO_CENSO"]
98     df[integer_columns] = df[integer_columns].astype("int32")
99     df["METRICA"] = df["METRICA"].astype("float")
100    return df
101
102
103 def save_dataframe(df: pd.DataFrame, indicador: str) -> None:
104    folder = f"./data/transformed/indicadores/{indicador}.parquet"
105    df.to_parquet(
106        folder,
107        engine="pyarrow",
108        compression="snappy",
109        index=None,
110        partition_cols=["NU_ANO_CENSO"]
111    )
```

4.4 Visualização

Foi criado um pacote *utils* para evitar repetição de código durante as consultas. Iniciando ele está a definição dos dicionários INDICADORES , que mapeia o nome do indicador para a sua respectiva tabela no DuckDB, e DIMENSOES e DIMENSOES_GEOGRAFICAS que mapeiam as dimensões para suas respectivas na tabela que contém os microdados (código 4.5, linha 13). Após isso, há definições de funções para facilitar a consulta no *DuckDB*. A função ***init_db_connection*** (código 4.5, linha 29), executada sempre que o pacote utils é importado, inicia uma conexão com o DuckDB, marca os dados armazenados em Parquet como tabelas para consulta e retorna uma variável contendo a conexão. Essa variável será utilizada em ***run_query*** (código 4.5, linha 43) que, dado um string de consulta, retorna um *Pandas Dataframe* com o resultado da consulta.

Listing 4.5: Pacote *utils*. Disponível em <https://github.com/rjribeiro/EduCensoExplorer/blob/main/app/utils.py>

```
1 INDICADORES = {
2     "Adequacao da Formacao Docente": "AFD",
3     "Percentual de Docentes com Curso Superior": "DSU",
```

```
    "Indice de Esforco Docente": "IED",
    "Media de Alunos por Turma": "ATU",
    "Media de Horas-aula Diaria": "HAD",
    "Taxas de Distorcao Idade-serie": "TDI",
    "Taxa de Aprovacao": "TAP",  # TRE
    "Taxa de Reprovacao": "TRP",  # TRE
    "Taxa de Abandono": "TAB"  # TRE
}

DIMENSOES = {
    "Dependência Administrativa": "TP_DEPENDENCIA",
    "Localidade": "TP_LOCALIZACAO",
}

DIMENSOES_GEOGRAFICAS = {
    "País": "NO_PAIS",
    "Região Geográfica": "NO_REGIAO",
    "Unidade da Federação": "NO_UF",
    "Mesorregião": "NO_MESORREGIAO",
    "Microrregião": "NO_MICRORREGIAO",
    "Município": "NO_MUNICIPIO"
}


@st.cache_resource
def init_db_connection() -> duckdb.DuckDBPyConnection:
    con = duckdb.connect()
    microdados = ds.dataset("data/transformed/microdados.parquet", format="parquet", partitioning="hive")
    con.register("microdados", microdados)

    indicadores = {}
    for indicador in INDICADORES.values():
        indicadores[indicador] = ds.dataset(f"data/transformed/indicadores/{indicador}.parquet", format="parquet",
                                            partitioning="hive")
        con.register(indicador, indicadores[indicador])
    return con


@st.cache_data
```

```
43 def run_query(query: str) -> pd.DataFrame:
44     return con.execute(query).df()
45 
46 con = init_db_connection()
```

As dashboards foram construídas utilizando Streamlit, ferramenta já apresentada em 2.3.4. Neste framework é necessário definir um arquivo Python por onde a aplicação começa a rodar. Esse arquivo pode ser conferido no Github do projeto [5]. É um script simples, que além de iniciar a aplicação, serve para definir o texto em *Markdown* que será mostrado na tela inicial. No mesmo diretório em que está o arquivo main tem um subdiretório chamado pages, e nele é onde deve ser colocado o código Python para cada aba da aplicação. Para esse trabalho serão 4 abas, cada uma delas sobre um métrica diferente: Acesso a serviços básicos, Quantidade de escolas, Quantidade matrículas e Indicadores educacionais.

Explicar o código de cada métrica seria repetitivo, pois todas seguem o mesmo padrão, com pequenas variações. Portanto, explicarei aqui a framework geral seguida. Os códigos de cada métrica podem ser acessados no Github do projeto [6].

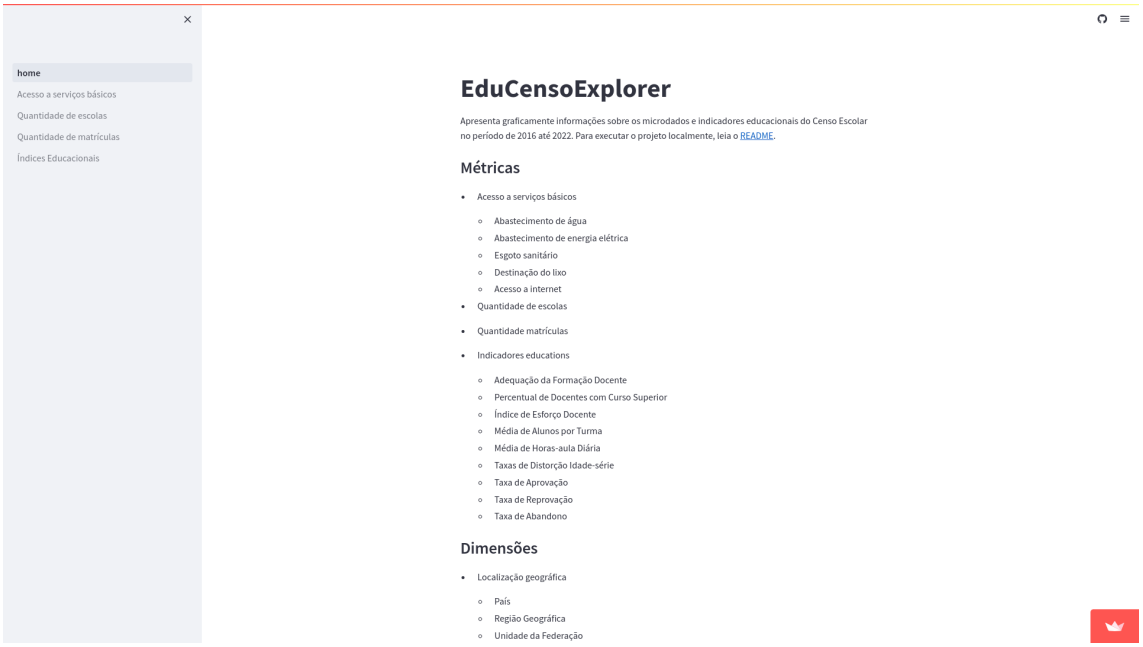
1. Definir tipo de gráfico cada métrica tem 2 escolhas possíveis: linha/barra ou linha/mapa.
2. Definir agregações: por tipo de localidade (urbana, rural, total), dependência administrativa (municipal, privada, estadual, federal, total) ou localidade geográfica (estado, município, etc).
3. Definir filtros: se o gráfico for linha/barra, o filtro será sobre localidade, dependência e localidade geográfica. Se o gráfico for mapa somente localidade, dependência.
4. Realizar consulta: utilizando a função *run_query* (código 4.5, linha 43)
5. Plotar gráfico

Nas proxímas cincos figuras há *prints* da aplicação, batizada de *EduCensoExplorer*.

[5] Disponível em <https://raw.githubusercontent.com/rjribeiro/EduCensoExplorer/main/app/home.py>

[6] Disponível em <https://github.com/rjribeiro/EduCensoExplorer/tree/main/app/pages>

Figura 4.3: Aplicativo *EduCensoExplorer*: Tela inicial.



Fonte: o autor

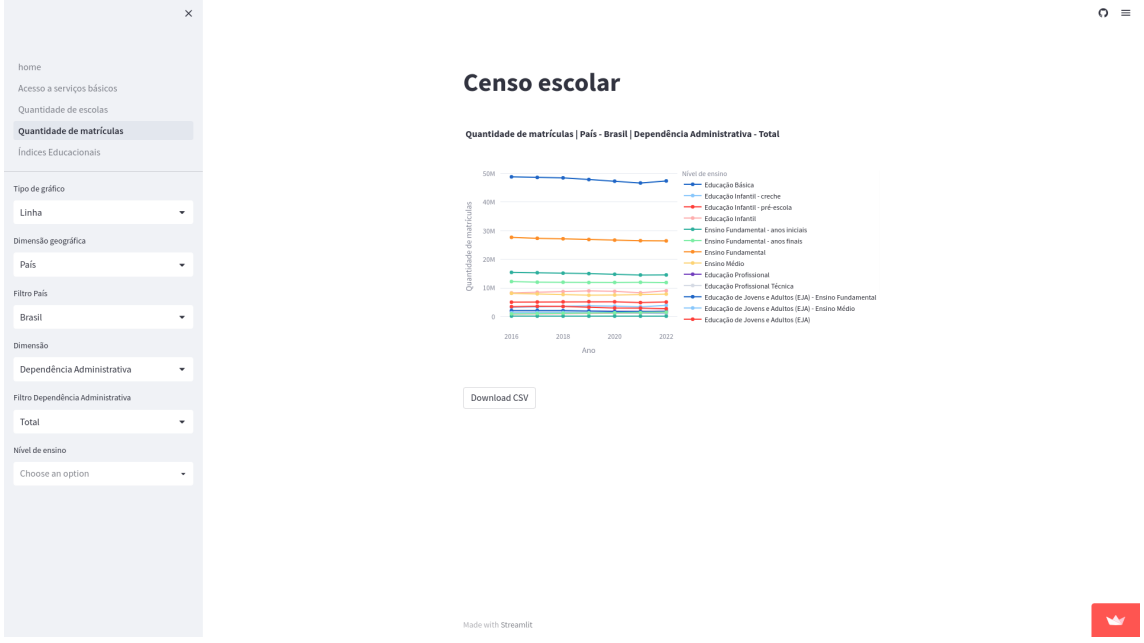
Figura 4.4: Aplicativo *EduCensoExplorer*: Tela "Acessos a serviços básicos".

Fonte: o autor

Figura 4.5: Aplicativo *EduCensoExplorer*: Tela "Quantidade de escolas".

Fonte: o autor

Figura 4.6: Aplicativo EduCensoExplorer: Tela "Quantidade de matrículas".



Fonte: o autor

Figura 4.7: Aplicativo *EduCensoExplorer*: Tela "Índices Educacionais".

Fonte: o autor

5 EXPERIMENTOS E RESULTADOS

Neste Capítulo estão apresentados os experimentos realizados e resultados para as diferentes funcionalidades da aplicação. Os *dashboards* podem ser acessados através da Streamlit Commnunity Cloud[1] ou através da máquina local, conforme explicado na página inicial do projeto no Github[2],

Para apresentação nesse capítulo, os gráficos tipo linha e barra tiveram os dados agrupados por país, enquanto que nos gráficos do tipo mapa foram agregados a nível de messoregião. Nesse nível o gráfico baixado é bem granulirazado e tem um boa visualização mesmo tendo o zoom padrão, que é o Brasil.

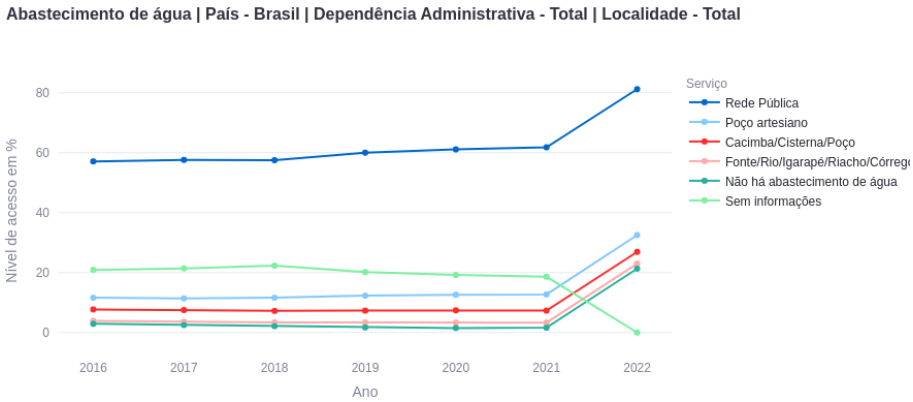
5.1 Acesso a serviços básicos

Os microdados do Censo Escolar informam a situação atual de cada escola em relação ao acesso de diversos serviços básicos. Para experimento neste trabalho foram analisados o avanço no acesso a rede pública de água potável, energia elétrica e tratamento de esgoto. Linha e Mapa são as opções de gráfico. Se escolhido mapa é necessário também escolher o subtipo de serviço.

Através das Figuras 5.1, 5.2 e 5.1 percebe-se que por muitos anos os níveis de acesso a rede pública dos serviços permaneram estagnados, mas que houve avanços no ano de 2022. Foi pesquisado sobre possíveis políticas públicas que possam ter contribuído para essa melhora, mas não foi possível chegar a nenhuma conclusão. Uma outra hipótese é que tenha faltado algum tratamento nos dados de 2022.

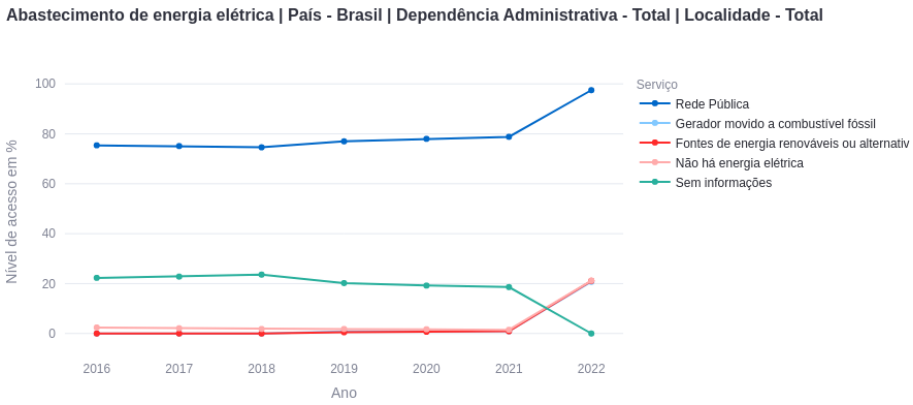
[1] Disponível em <ttps://educensoexplorer.streamlit.app/>
[2] Disponível em <https://github.com/rjribeiro/EduCensoExplorer>

Figura 5.1: Acesso a abastecimento de água nas escolas do Brasil



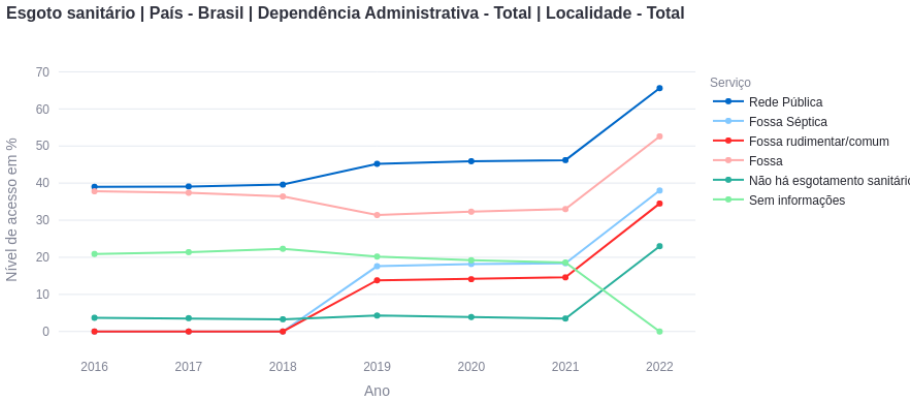
Fonte: elaboração própria com dados do Censo Escolar

Figura 5.2: Acesso a abastecimento de energia elétrica nas escolas do Brasil



Fonte: elaboração própria com dados do Censo Escolar

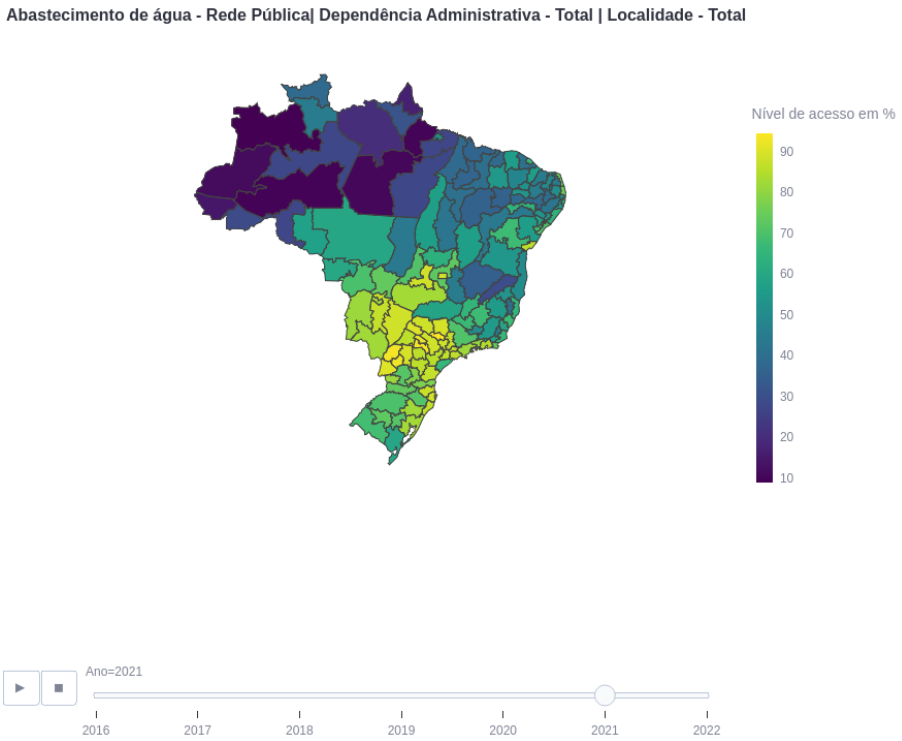
Figura 5.3: Acesso a esgoto sanitário nas escolas do Brasil



Fonte: elaboração própria com dados do Censo Escolar

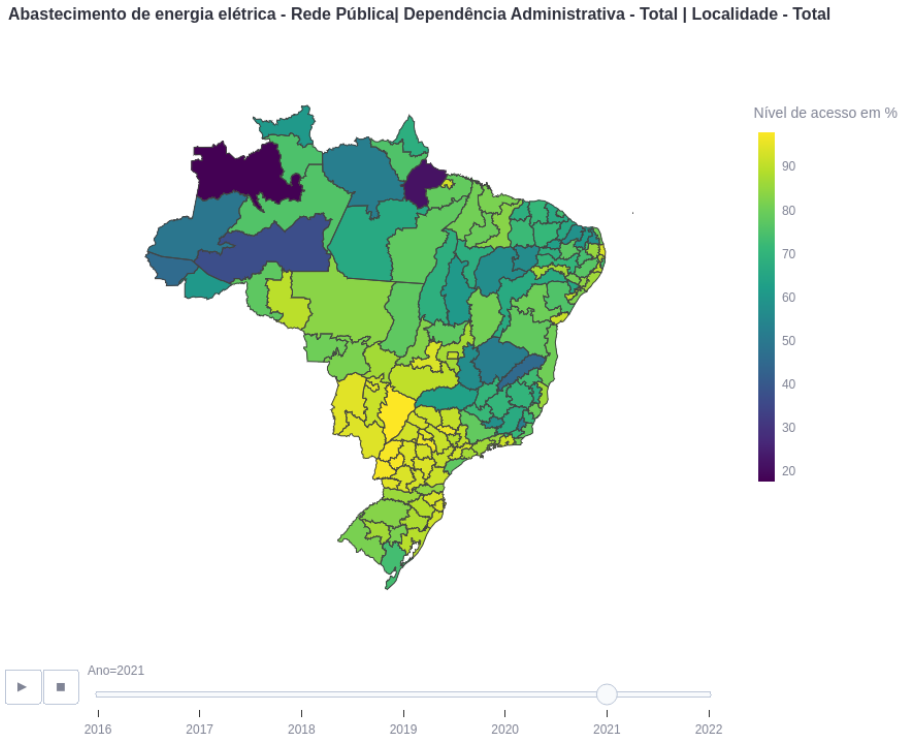
Otipo mapa é utilizado para buscar relações geográficas no acesso a serviços básicos. Na aplicação há um *slider widget* que realiza a transição do plot de acordo com o ano. As figuras 5.4, 5.5 e 5.6 ilustram mapas que indicam maior presença da rede pública de serviços básicos na região geoeconômica Centro-Sul.

Figura 5.4: Acesso a abastecimento de água rede pública nas escolas do Brasil em 2022 agrupado por mesorregião



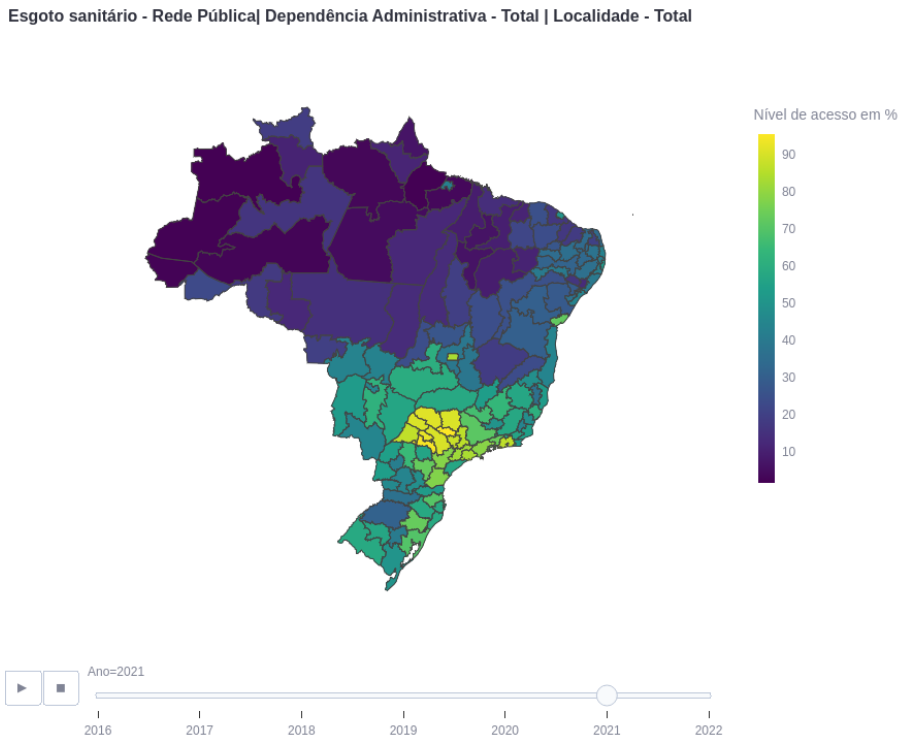
Fonte: elaboração própria com dados do Censo Escolar

Figura 5.5: Acesso a abastecimento de energia elétrica na rede pública nas escolas do Brasil em 2022 agrupado por mesorregião



Fonte: elaboração própria com dados do Censo Escolar

Figura 5.6: Acesso a esgoto sanitário rede pública nas escolas do Brasil em 2022 agrupado por mesorregião

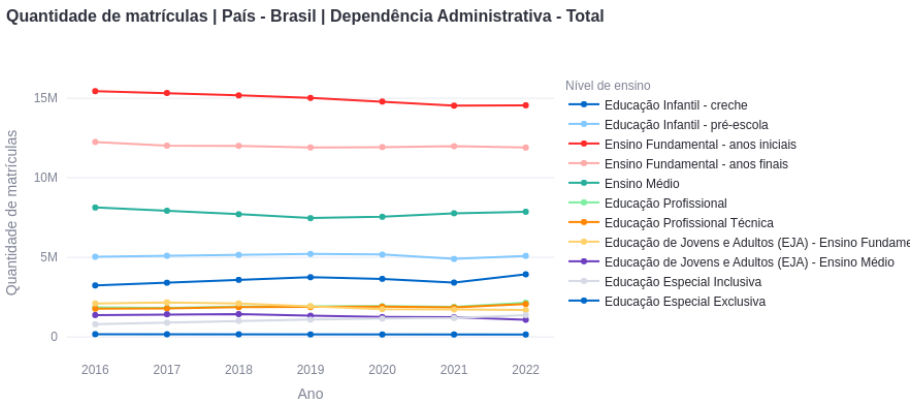


Fonte: elaboração própria com dados do Censo Escolar

5.2 Quantidade de escolas e matrículas

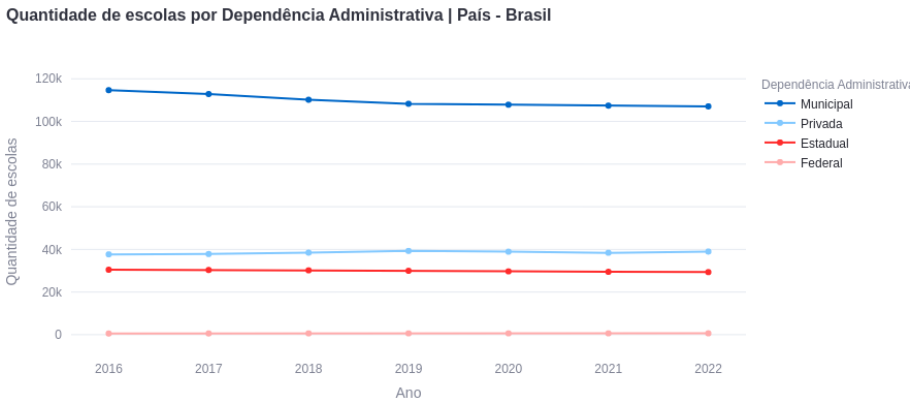
Na aba "Quantidade de escolas"e "Quantidade de matrículas", há linhas e barras como tipos de gráficos, sendo possível agrupar por dependência administrativa ou localidade, e filtrar por país, unidade federativa, mesorregião, microrregião e município. Nas matrículas, também é possível filtrar por um ou mais níveis de ensino. As informações apresentadas pelos gráficos referentes às figuras 5.7 e 5.8 se mostraram de baixa utilidade. Seria mais informativo se fosse possível agregar os dados por dependência administrativa e localidade.

Figura 5.7: Quantidade de matrículas no Brasil - Dependência Administrativa total



Fonte: elaboração própria com dados do Censo Escolar

Figura 5.8: Quantidade de escolas no Brasil - Dependência Administrativa total

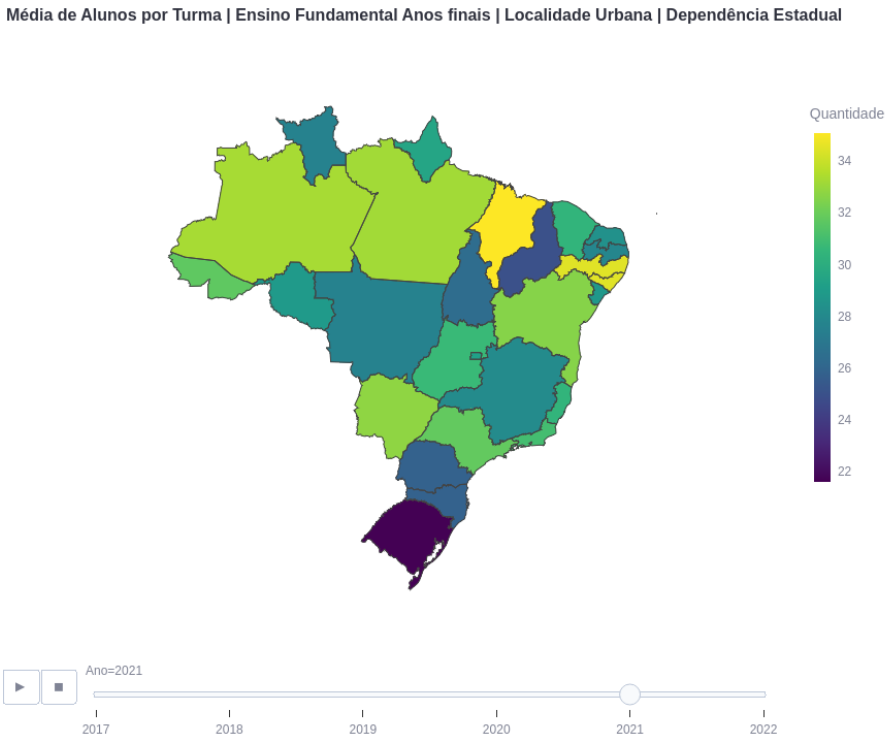


Fonte: elaboração própria com dados do Censo Escolar

5.3 Índices Educacionais

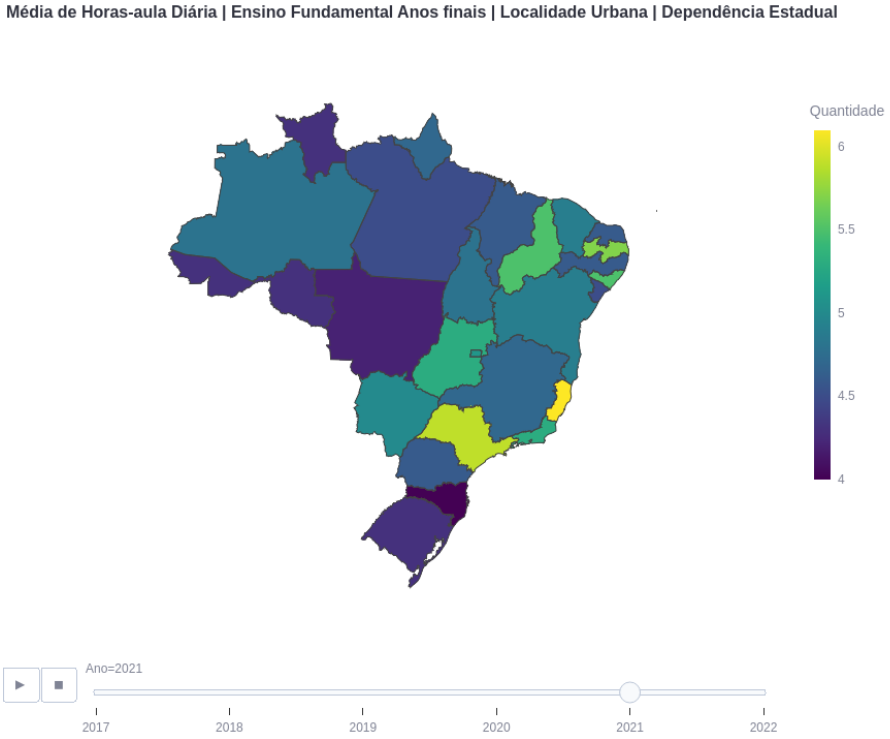
A base de indicadores não possui dados a nível de mesorregião. Devido a isso, os plots geográficos aqui mostrados serão a nível de unidade federativa. Para os indicadores divididos em grupo/nível, foram selecionados aqueles com maior prevalência. Como filtros, foram selecionadas localidades urbanas, dependência estadual, e em nível de ensino foi definido "ensino fundamental - anos finais". Para o ano, foi escolhido o ano de 2021, que é o ano mais recente em que estão presentes todos os indicadores. Foram escolhidos os indicadores Média de alunos por turma (Figura 5.9), Média de horas-aula diárias (Figura 5.10), Adequação da formação docente (Figura 5.11) e Índice de esforço docente (Figura 5.12) com objetivo de explicar Distorção idade-série (Figura 5.13) e Taxa de abandono (Figura 5.14). Visualmente, definir essa relação não foi possível. Entre os motivos para isso, pode ser que a granularidade geográfica pode ser muito alta, sendo necessário fazer uma análise municipal. Essa análise demonstrou precisar de uma etapa de transformação adicional para definir a unidade federativa dos municípios. Sem esse filtro, o plot fica com uma escala de cores pouco variada para municípios próximos entre si, conforme mostrado na Figura 5.15, que, além disso, também ajuda a demonstrar outro problema: há municípios em branco. Isso pode ser devido à ausência real do indicador ou à chave utilizada para fazer a junção entre o dataset de indicadores e o GEOJSON municipal, que é o nome do município, apresentou inconsistências entre as tabelas. Se a junção for o problema, isso pode ser resolvido utilizando o código IBGE do município, que está no GEOJSON, mas não nos indicadores, sendo necessário um processo adicional na sua transformação para conseguir esse dado.

Figura 5.9: Média de alunos por turma



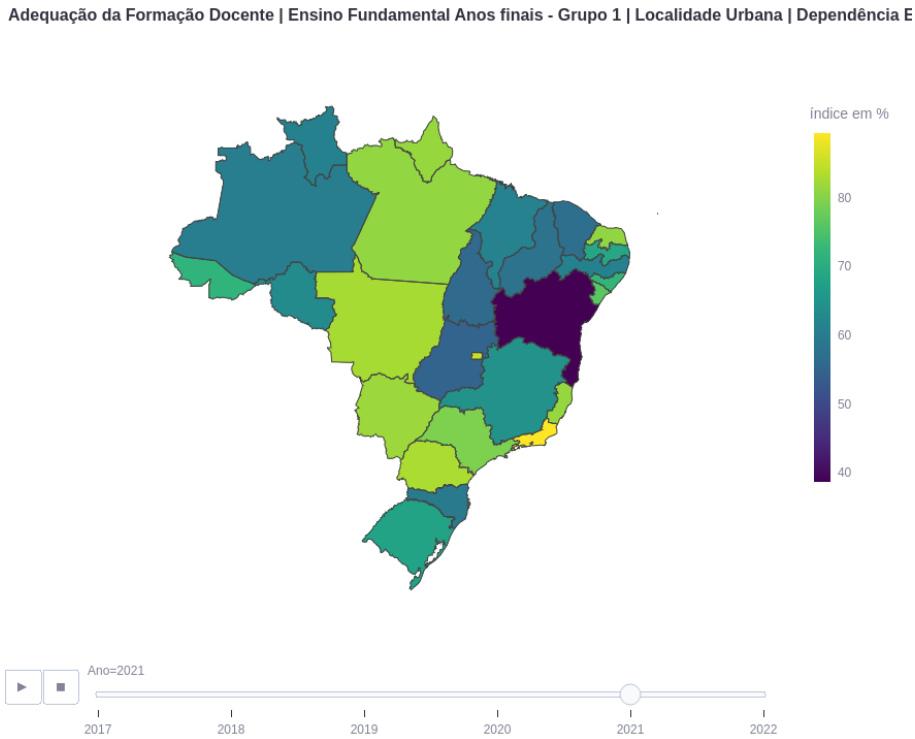
Fonte: elaboração própria com dados do Censo Escolar

Figura 5.10: Média de horas-aula diárias



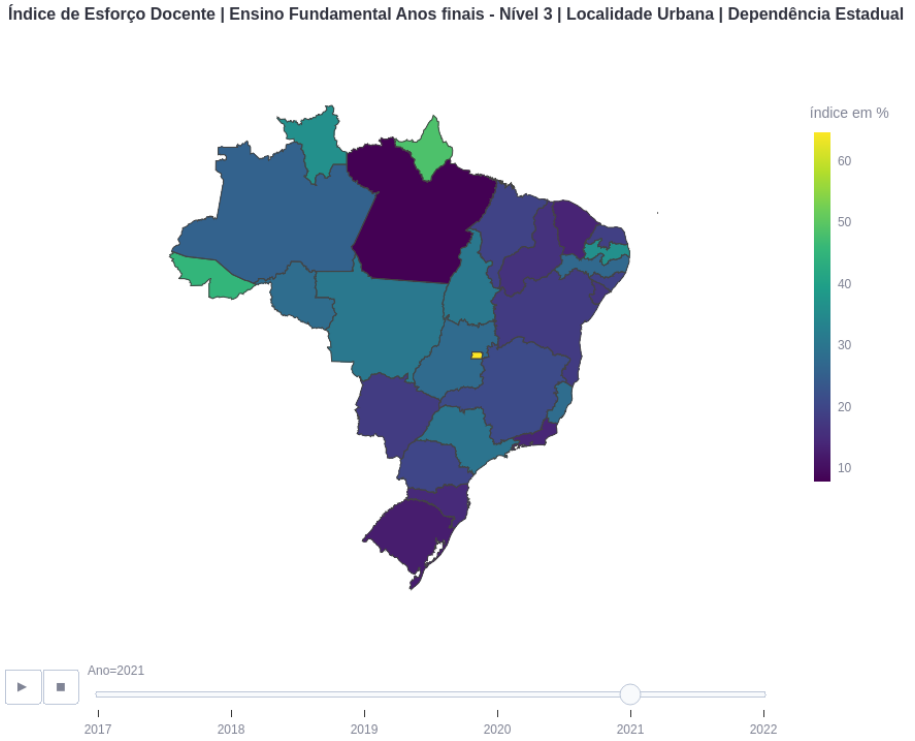
Fonte: elaboração própria com dados do Censo Escolar

Figura 5.11: Adequação da formação docente



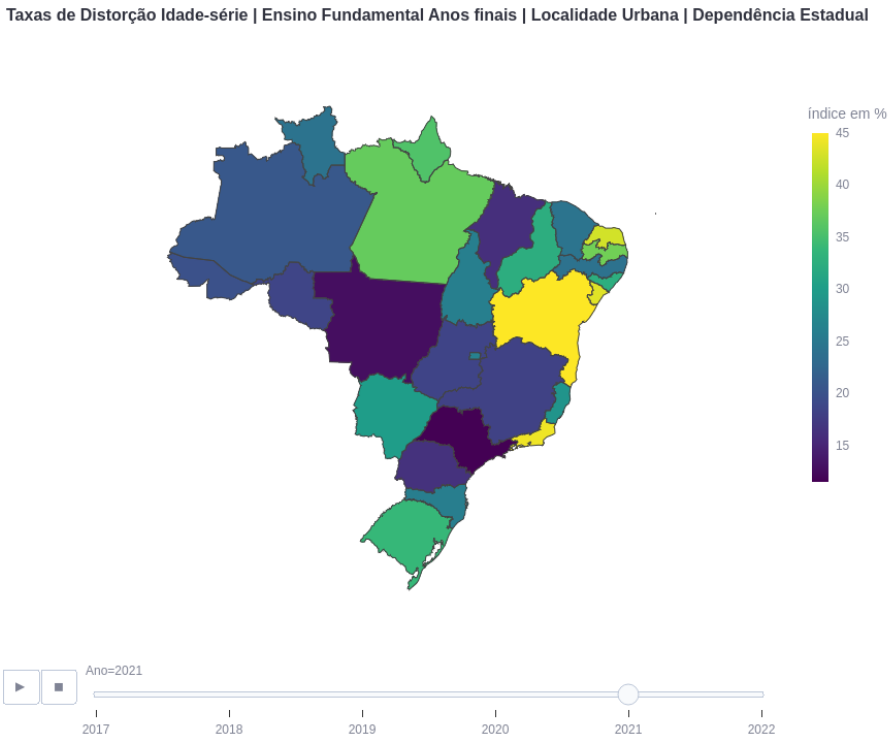
Fonte: elaboração própria com dados do Censo Escolar

Figura 5.12: Índice de esforço docente



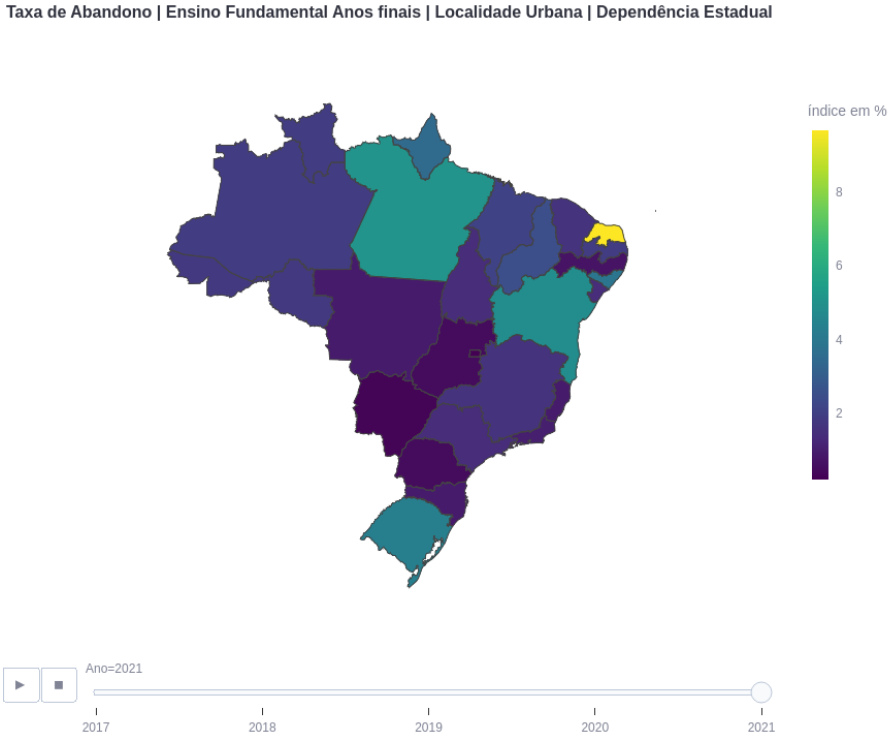
Fonte: elaboração própria com dados do Censo Escolar

Figura 5.13: Distorção idade-série



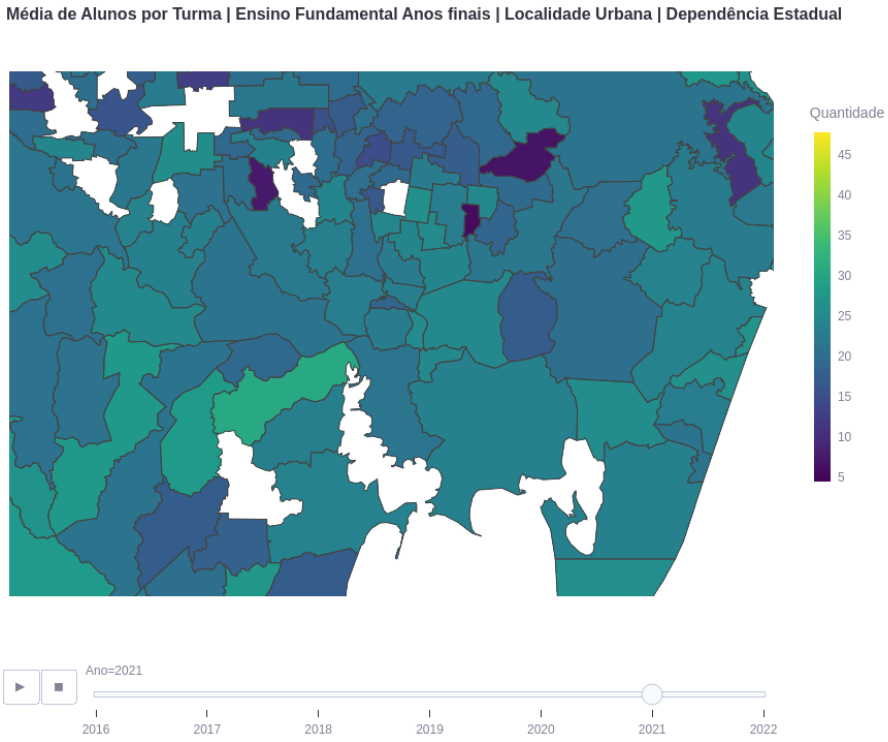
Fonte: elaboração própria com dados do Censo Escolar

Figura 5.14: Taxa de abandono



Fonte: elaboração própria com dados do Censo Escolar

Figura 5.15: Média de alunos por turma com *zoom* no nordeste do Rio Grande Sul, com municípios da serra, capital e litoral



Fonte: elaboração própria com dados do Censo Escolar

6 CONCLUSÃO

O propósito do trabalho era definir um processo que integre conceitos e ferramentas tecnológicas em uma aplicação de dados que auxilie pesquisadores na busca de respostas para o ensino educacional. Com esse objetivo em mente, o trabalho foi elaborado de forma a ser amplamente compartilhado e facilmente extendido a qualquer outro projeto sobre o Censo Escolar. Graças ao formato de arquivo Parquet e a definição de schema mais eficiente e econômica, os dados tratados puderam ser compactados e compartilhados diretamente no GitHub do projeto, sem a necessidade de recorrer a opções externas de armazenamento. Além disso, o Streamlit facilita a integração do repositório com a aplicação web no Streamlit Cloud, eliminando processos complexos de integração e deploy contínuos e tornando trivial o acesso público à aplicação. Todas as ferramentas utilizadas no trabalho são de código aberto ou gratuitas, com documentação de alta qualidade disponível gratuitamente na *Web*, sem a necessidade de recorrer a cursos pagos para aprender a lidar com os recursos tecnológicos utilizados pelo projeto.

Grande parte da dificuldade em relação aos dados brutos do Censo Escolar está na forma como são compartilhados pelo INEP. O formato de arquivo Excel utilizado na base de indicadores dificulta bastante o tratamento dos dados de maneira automatizada. Seria conveniente unificar o conjunto de dados de municípios com o conjunto que contém dados dos estados, regiões e do Brasil. Quanto ao conjunto de dados dos microdados, é importante que ele atenda aos requisitos da LGPD, mas sem remover informações ao nível do aluno, que são necessárias para uma análise mais abrangente e eficaz. Para isso, é necessário promover uma ampla discussão sobre a privacidade dos dados abertos, envolvendo entidades públicas, ONGs, pesquisadores e a população em geral, sem recorrer a decisões monocráticas que possam prejudicar o desenvolvimento de pesquisas em andamento ou a reprodutibilidade de pesquisas já concluídas com dados antigos. Em relação à visualização de dados, os *dashboards* podem ser adaptadas conforme a necessidade de pesquisa, tendo como guias os implementadas neste trabalho.

Em geral, o propósito do trabalho foi atingido, com uma estrutura para

ETL e visualização robusta, econômica e de fácil execução. Para trabalhos futuros, sugere-se extender o framework aqui descrito para pesquisas envolvendo coleta e transformação não só de dados do Censo Escolar, mas como qualquer conjunto de dados abertos, com finalidade não somente de visualização de dados, mas também mineração de dados e *Machine Learning*.

REFERÊNCIAS

APACHE.ORG. **Apache Parquet**. 2023. Disponível na Internet: <https://parquet.apache.org/>.

BARROS, T.; SILVA, I.; GUEDES, L. A. Modelagem e visualização científica de dados educacionais: Estudo de caso sobre o desempenho em componentes curriculares. Em: . [S.l.: s.n.], 2017. p. 654.

BERNERS-LEE, T. **Linked Data**. 2006. Disponível na Internet: <https://www.w3.org/DesignIssues/LinkedData.html>.

BRASIL. **Senado 454-22**. 2022. Disponível na Internet: <https://www25.senado.leg.br/web/atividade/materias/-/materia/153130>.

BRASIL. **Censo Escolar**. 2023. Disponível na Internet: <https://www.gov.br/inep/pt-br/areas-de-atuacao/pesquisas-estatisticas-e-indicadores/censo-escolar/resultados>.

BRASIL. **Indicadores Educacionais**. 2023. Disponível na Internet: <https://www.gov.br/inep/pt-br/acesso-a-informacao/dados-abertos/indicadores-educacionais>.

BRASIL. **Nota técnica Adequação da Formação Docente**. 2023. Disponível na Internet: <https://download.inep.gov.br/informacoes_estatisticas/indicadores_educacionais/2014/docente_formacao_legal/nota_tecnica_indicador_docente_formacao_legal.pdf>.

BRASIL. **Nota técnica Índice de Esforço Docente**. 2023. Disponível na Internet: <https://download.inep.gov.br/informacoes_estatisticas/indicadores_educacionais/2014/docente_esforco/nota_tecnica_indicador_docente_esforco.pdf>.

DATE, C. J. **A Guide to the SQL Standard**. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 1989.

DUCKDB.ORG. **DuckDB Website**. 2023. Disponível na Internet: <https://duckdb.org/>.

EL-SAPPAGH, S. H. A.; HENDAWI, A. M. A.; El Bastawissy, A. H. A proposed model for data warehouse etl processes. **Journal of King Saud University - Computer and Information Sciences**, v. 23, n. 2, p. 91–104, 2011. ISSN 1319-1578. Disponível na Internet: <https://www.sciencedirect.com/science/article/pii/S131915781100019X>.

FERREIRA, L.; RODRIGUES, R.; SOUZA, R. Dados abertos educacionais brasileiros: Um mapeamento sistemático da literatura. Em: **Anais do XXXII Simpósio Brasileiro de Informática na Educação**. Porto Alegre, RS, Brasil: SBC, 2021. p. 1186–1195. ISSN 0000-0000. Disponível na Internet: <https://sol.sbc.org.br/index.php/sbie/article/view/18141>.

ISOTANI, S.; BITTENCOURT, I. **Dados Abertos Conectados: em Busca da Web do Conhecimento**. [S.l.: s.n.], 2015. ISBN 978-85-7522-449-6.

IVANOV, T.; PERGOLESI, M. The impact of columnar file formats on sql-on-hadoop engine performance: A study on orc and parquet. **Concurrency and Computation: Practice and Experience**, v. 32, n. 5, p. e5523, 2020. E5523 cpe.5523. Disponível na Internet: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5523>.

JANSSEN, M.; CHARALABIDIS, Y.; ZUIDERWIJK, A. Benefits, adoption barriers and myths of open data and open government. **Information Systems Management**, Taylor Francis, v. 29, n. 4, p. 258–268, 2012. Disponível na Internet: <https://doi.org/10.1080/10580530.2012.716740>.

KHORASANI, M.; ABDOU, M.; FERNÁNDEZ, J. H. Getting started with streamlit. Em: _____. **Web Application Development with Streamlit: Develop and Deploy Secure and Scalable Web Applications to the Cloud Using a Pure Python Framework**. Berkeley, CA: Apress, 2022. p. 1–30. ISBN 978-1-4842-8111-6. Disponível na Internet: <https://doi.org/10.1007/978-1-4842-8111-6_1>.

LGPD-BRASIL. **Inep oculta informações do sistema e deleta microdados do Censo Escolar e Enem**. 2022. Disponível na Internet: <"https://www.lgpdbrasil.com.br/inep-oculta-informacoes-do-sistema-e-deleta-microdados-do-censo-escolar-e-enem/">.

MAHMUD, M. S.; HUANG, J. Z.; SALLOUM, S.; EMARA, T. Z.; SADATDIYNOV, K. A survey of data partitioning and sampling methods to support big data analysis. **Big Data Mining and Analytics**, v. 3, n. 2, p. 85–101, 2020.

OLIVEIRA, L. F. B. de; SOARES, S. S. D. **Determinantes da repetência escolar no Brasil: uma análise de painel dos censos escolares entre 2007 e 2010**. 2012. Disponível na Internet: <https://repositorio.ipea.gov.br/handle/11058/1262>.

OLIVEIRA, L. F. B. de; SOARES, S. S. D. **O Impacto do Programa Bolsa Família Sobre a Repetência: Resultados a partir do Cadastro Único, Projeto Frequência e Censo Escolar**. [S.l.], 2013. Disponível na Internet: <https://ideas.repec.org/p/ipc/opport/233.html>.

OPENDATAHANDBOOK.ORG. **Open Data Handbook**. 2023. Disponível na Internet: <https://opendatahandbook.org/guide/pt_BR/>.

PYTHON.ORG. **Python Website**. 2023. Disponível na Internet: <https://www.python.org/>.

RAASVELDT, M.; MüHLEISEN, H. Duckdb: An embeddable analytical database. Em: **Proceedings of the 2019 International Conference on Management of Data**. New York, NY, USA: Association for Computing Machinery, 2019. (SIGMOD '19), p. 1981–1984. ISBN 9781450356435. Disponível na Internet: <https://doi.org/10.1145/3299869.3320212>.

RAMALHO, L. **Fluent python**. [S.l.]: "O'Reilly Media, Inc.", 2022.

SILVA, M. X. G. da; LOPES, S. d. F.; PEREIRA, D. d. S. Que indicadores influenciam na qualidade da educação da paraíba? what indicators influence the quality of education in paraíba? **Brazilian Journal of Development**, v. 8, n. 10, p. 66943–66959, Oct. 2022. Disponível na Internet: <https://ojs.brazilianjournals.com.br/ojs/index.php/BRJD/article/view/53058>.

SILVA, Y. N.; ALMEIDA, I.; QUEIROZ, M. Sql: From traditional databases to big data. Em: **Proceedings of the 47th ACM Technical Symposium on Computing Science Education**. New York, NY, USA: Association for Computing Machinery, 2016. (SIGCSE '16), p. 413–418. ISBN 9781450336857. Disponível na Internet: <https://doi.org/10.1145/2839509.2844560>.

STANčIN, I.; JOVIć, A. An overview and comparison of free python libraries for data mining and big data analysis. Em: **2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)**. [S.l.: s.n.], 2019. p. 977–982.

STREAMLIT.IO. **Streamlit Website**. 2023. Disponível na Internet: <https://streamlit.io/gallery>.

TIOBE.COM. **TIOBE Index**. 2023. Disponível na Internet: <https://www.tiobe.com/tiobe-index/>.

VASSILIADIS, P.; SIMITSIS, A. Near real time etl. Em: _____. **New Trends in Data Warehousing and Data Analysis**. Boston, MA: Springer US, 2009. p. 24. ISBN 978-0-387-87431-9. Disponível na Internet: <https://doi.org/10.1007/978-0-387-87431-9_2>.