UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

GUILHERME RIBEIRO COSTA CARVALHO

BOAS PRÁTICAS PARA A CONSTRUÇÃO DE PIPELINES DE DADOS EM PEQUENAS E MÉDIAS EMPRESAS: UM ESTUDO APLICADO DE PROCESSOS ETL

RECIFE
2025

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Carvalho, Guilherme Ribeiro Costa.
Boas práticas para a construção de pipelines de dados em pequenas e médias empresas: um estudo aplicado de processos ETL / Guilherme Ribeiro Costa Carvalho. - Recife, 2025.
70 p. : il., tab.

Orientador(a): Adiel Teixeira de Almeida Filho
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Sistemas de Informação - Bacharelado, 2025.
Inclui referências, apêndices.

1. pipelines de dados. 2. ETL. 3. PMEs. 4. pequenas e médias empresas. 5. boas práticas. 6. orquestração. I. Almeida Filho, Adiel Teixeira de . (Orientação). II. Título.

000 CDD (22.ed.)

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

GUILHERME RIBEIRO COSTA CARVALHO

BOAS PRÁTICAS PARA A CONSTRUÇÃO DE PIPELINES DE DADOS EM PEQUENAS E MÉDIAS EMPRESAS: UM ESTUDO APLICADO DE PROCESSOS ETL

Monografia apresentada ao Centro de Informática (CIn) da Universidade Federal de Pernambuco (UFPE), como requisito para conclusão do Curso de Sistemas de Informação, orientada pelo professor Adiel Teixeira de Almeida Filho.

Aprovado em: 30/07/2025

BANCA EXAMINADORA

Prof. Adiel Teixeira de Almeida Filho (Orientador)
Universidade Federal de Pernambuco

Profa. Juscimara Gomes Avelino (Examinadora Interna)
Universidade Federal de Pernambuco

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, por não medirem esforços, desde sempre, para que eu tivesse uma boa educação e um bom futuro. Agradeço aos meus irmãos, por todo o companheirismo e apoio durante tantos anos, seja nos momentos bons ou ruins.

Agradeço aos meus amigos que construí ao longo da faculdade, que foram essenciais nesses quatro anos. Em especial, Breno, Luca, Maria Luiza, Wilson, Vinícius e Rodrigo. Os vários trabalhos em grupo, desafios compartilhados e momentos bons vividos juntos tornaram essa jornada mais tranquila.

Também sou grato aos meus amigos mais antigos, que permanecem presentes mesmo com o passar do tempo. Em especial ao grupo Casa Bolada e à minha amiga Luíza Neves, obrigado pelo apoio diário e constante e pela amizade genuína, que sempre fizeram a diferença.

RESUMO

Este trabalho busca identificar e organizar boas práticas e ferramentas para a construção de pipelines de dados em pequenas e médias empresas (PMEs), com foco em processos de ETL (Extract, Transform, Load). A abordagem utilizada foi qualitativa, de natureza exploratória e descritiva, combinando uma Pesquisa Bibliográfica com a implementação de uma Prova de Conceito (PoC) em ambiente local simplificado. A pesquisa destacou a modularidade das etapas, automação por orquestração e rastreabilidade como pilares centrais para a construção de pipelines eficientes. A PoC utilizou ferramentas acessíveis como Python, PostgreSQL, Docker e Apache Airflow, integrando dados de um cenário simulado de e-commerce para validar um pipeline modular e orquestrado. Também foi adotado o modelo dimensional em estrela para facilitar análises. Os resultados mostraram que empresas pequenas e médias podem aplicar estas práticas mesmo com poucos recursos, desde que escolham as tecnologias certas. Embora o teste tenha usado dados fictícios, as descobertas sugerem que o método pode funcionar bem em situações reais, ajudando empresas a melhorarem suas análises de dados.

Palavras-chave: pipelines de dados; ETL; PMEs; pequenas e médias empresas; boas práticas; orquestração.

ABSTRACT

This study aims to identify and organize best practices and tools for building data pipelines in small and medium-sized enterprises (SMEs), with a focus on ETL (Extract, Transform, Load) processes. The approach used was qualitative, with an exploratory and descriptive nature, combining a Systematic Literature Review (SLR) with the implementation of a Proof of Concept (PoC) in a simplified local environment. The review highlighted the modularity of pipeline stages, orchestration-based automation, and traceability as key pillars for building efficient pipelines. The PoC used accessible tools such as Python, PostgreSQL, Docker, and Apache Airflow, integrating data from a simulated e-commerce scenario to validate a modular and orchestrated pipeline. A star schema dimensional model was also adopted to support analytical queries. The results showed that small and medium-sized businesses can apply these practices even with limited resources, provided that the right technologies are chosen. Although the test used fictional data, the findings suggest that the method can work well in real-world situations, helping companies improve their data analysis capabilities.

Key words: data pipelines; ETL; SMEs; small and medium enterprises; best practices; orchestration.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

PMEs.....Pequenas e Médias Empresas
PoC..... Prova de Conceito
ETL.....Extract, Transform and Load (Extração, Transformação e Carregamento)

SUMÁRIO

1. INTRODUÇÃO

A crescente geração de informações nas últimas décadas tem impulsionado empresas de todos os tamanhos a buscar formas cada vez mais eficientes e sofisticadas de coletar, transformar e analisar dados, com o objetivo de apoiar decisões estratégicas e obter vantagem competitiva no mercado. O volume, a velocidade e a variedade dos dados disponíveis atualmente representam tanto um desafio quanto uma oportunidade para as organizações que desejam extrair valor significativo dessas informações (BECKER et al., 2023).

Nesse contexto, pipelines de dados e processos ETL tornaram-se essenciais para organizar e analisar a vasta quantidade de informações. Eles realizam coletas através de várias fontes e as transformam em conhecimentos úteis para empresas. Enquanto empresas grandes já usam regularmente essas tecnologias, as PMEs têm dificuldade para implementá-las devido a limitações técnicas, financeiras e operacionais (JOHNSON et al., 2024; ALSIBHAWI et al., 2023).

As PMEs brasileiras, responsáveis por uma significativa parcela do PIB nacional e pela geração de empregos formais (SEBRAE, 2018 apud BECKER et al., 2023), estão inseridas em um mercado muito competitivo. Para essas empresas, a capacidade de extrair informações úteis a partir de grandes volumes de dados é um diferencial decisivo. Contudo, problemas como escassez de investimentos, carência de profissionais qualificados e resistência cultural à adoção de soluções tecnológicas emergentes ainda dificultam esse avanço.

Chikhalkar et al. (2025) destacam que a ausência de frameworks facilmente adaptáveis às necessidades específicas das PMEs representa um dos principais entraves à implementação e manutenção de pipelines de dados escaláveis e sustentáveis. Alsibhawi et al. (2023) reforçam a importância de modelos conceituais claros e bem estruturados para orientar a adoção de soluções de Business Intelligence e ETL em ambientes com recursos limitados.

Outro desafio importante diz respeito à complexidade e heterogeneidade dos dados, que exigem arquiteturas flexíveis e adaptáveis, frequentemente ausentes nas PMEs (MEHMOOD; ANEES, 2020). A integração de dados provenientes de fontes como redes sociais, sistemas legados e plataformas de e-commerce, com diferentes formatos, velocidades e volumes, demanda pipelines robustos e contínuos, alinhados a um cenário de Big Data (NWOKEJI; MATOVU, 2021).

Do ponto de vista técnico, o custo elevado das ferramentas tradicionais e a falta de suporte para dados não estruturados limitam a escalabilidade das soluções (MEHMOOD; ANEES, 2020). Além disso, quando faltam regras claras para gerenciar dados e processos padronizados de ETL, as informações ficam menos confiáveis, causando erros e perda de oportunidades de negócio (TAWIL et al., 2024). Estes problemas não só dificultam a automação e a qualidade dos dados, mas também impedem que as empresas avancem digitalmente e melhorem suas análises.

Dessa forma, o contexto atual das PMEs revela que: ao mesmo tempo em que os dados se tornam um ativo estratégico cada vez mais valioso, a implementação de pipelines e processos ETL permanece restrita. A partir desse diagnóstico, este trabalho parte do ponto de um problema recorrente: a ausência de diretrizes claras, acessíveis e aplicáveis para a construção de pipelines de dados eficazes em PMEs. Com base em uma Pesquisa Bibliográfica, o objetivo é mapear e analisar os principais desafios, boas práticas, ferramentas e abordagens utilizadas, construindo uma base sólida de conhecimento que possa orientar futuras implementações em contextos empresariais similares.

2. MOTIVAÇÃO E JUSTIFICATIVA

A adoção de pipelines de dados eficientes tem se consolidado como fator estratégico para a tomada de decisões nas organizações, impulsionando inovação e competitividade (BECKER et al., 2023). No entanto, PMEs enfrentam barreiras significativas para implementar soluções baseadas em dados, especialmente pela escassez de recursos técnicos e humanos (JOHNSON et al., 2024; ALSIBHAWI et al., 2023).

Mehmood e Anees (2020) destacam que a implementação e manutenção de processos ETL em ambientes com infraestrutura restrita são tarefas complexas. Além disso, Rozony et al. (2024) apontam que a falta de padronização, as diferenças semânticas entre fontes e as limitações na integração em tempo real dificultam ainda mais o aproveitamento dos dados.

Soluções existentes tendem a ser projetadas para grandes corporações, o que muitas vezes as torna inviáveis para PMEs devido à sua complexidade e custo (CHIKHALKAR et al., 2025). Embora arquiteturas híbridas baseadas em otimização inteligente sejam alternativas promissoras, essas abordagens ainda são pouco conhecidas por pequenas empresas (DINESH; DEVI, 2024).

Estudos indicam que há uma lacuna significativa na literatura quanto à sistematização de boas práticas adaptadas especificamente às PMEs (NWOKEJI; MATOVU, 2021). Além

disso, a ausência de uma cultura orientada a dados e de políticas eficazes de governança compromete a qualidade, a segurança e a escalabilidade das soluções adotadas.

Diante desse cenário, esta pesquisa busca consolidar práticas, ferramentas e soluções viáveis que considerem as limitações e potencialidades das PMEs. Ao propor uma prova de conceito aplicada, o estudo visa fomentar a democratização do acesso às tecnologias de dados, promovendo maior resiliência e competitividade no mercado contemporâneo.

3. OBJETIVOS

3.1 Objetivo geral

Mapear e sistematizar boas práticas, ferramentas, arquiteturas e desafios técnicos para a construção de pipelines de dados em PMEs, com foco em processos ETL, a fim de oferecer recursos práticos e estratégicos para adoção de soluções escaláveis e alinhadas à realidade dessas organizações.

3.2 Objetivos específicos

- Identificar os principais desafios técnicos e gerenciais na adoção de pipelines e processos ETL em PMEs.
- Mapear ferramentas, frameworks e abordagens recomendadas para implementação de pipelines em PMEs.
- Sistematizar boas práticas e diretrizes adaptáveis à realidade operacional das PMEs.
- Analisar os benefícios estratégicos do uso de pipelines e ETL.
- Elaborar uma síntese crítica que sirva como base para futuras implementações.

4. FUNDAMENTAÇÃO TEÓRICA

4.1 Definição e Caracterização das PMEs

A definição de PMEs é ampla e não existe um padrão universalmente aceito, conforme discutido por Berisha e Pula (2015). Cada país usa critérios diferentes para classificar essas empresas. Esses critérios incluem número de funcionários, quanto a empresa fatura por ano e valor total dos seus bens.

Na União Europeia, a definição oficial considera empresas médias aquelas com menos de 250 empregados, pequenas com menos de 50 e microempresas com menos de 10

funcionários. Além do critério do número de funcionários, também são utilizados indicadores financeiros, como faturamento anual ou balanço patrimonial total, como apresentado na Tabela 1.

Tabela 1: Definição de PMEs segundo critérios da União Europeia.

Enterprise category	Headcount: Annual Work Unit (AWU)	Annual turnover	or	Annual balance sheet total
Medium-sized	< 250	≤ €50 million	or	≤ €50 million
Small	< 50	≤ €10 million	or	≤ €10 million
Micro	< 10	≤ €2 million	or	≤ €2 million

Fonte: European Commission (2005), adaptado de Berisha e Pula (2015).

Já o Banco Mundial adota uma abordagem que também considera ativos totais e vendas anuais, além do número de empregados, como mostrado na Tabela 2.

Tabela 2: Definição de PMEs segundo critérios do Banco Mundial.

Enterprise category	Number of employees	Total assets	or	Total annual sales
Medium	> 50; ≤ 300	>$3,000,000; ≤ $15,000,000	or	> $3,000,000; ≤ $15,000,000
Small	> 10; ≤ 50	> $100,000; ≤ $3,000,000	or	> $100,000; ≤ $3,000,000
Micro	< 10	≤ $100,000	or	≤ $100,000

Fonte: Independent Evaluation Group (2008), adaptado de Berisha e Pula (2015).

Além disso, os critérios variam muito de um país para outro. A OECD mostra uma comparação entre vários países usando o número de funcionários, como podemos ver na Tabela 3.

Tabela 3: Distribuição das empresas por número de empregados em diferentes países.

	Micro	Small	Medium	SME	Large
EU countries, Iceland, Norway, and Switzerland	1 – 9	10 – 49	50 – 249	1 – 249	250+
Australia	0 – 9	10 – 49	50 – 199	0 –199	200+
Canada	0 – 9	10 – 49	50 – 499	0 – 499	500+

Japan	4 – 9	10 – 49	50 – 249	1 – 249	250+
Korea	5 – 9	10 – 49	50 – 199	5 – 199	200+
Mexico	0 – 10	11 – 50	51 – 250	1 – 250	251+
New Zealand	1 – 9	10 – 49	50 – 99	0 – 99	100+
Turkey	1 – 19	20 – 49	50 – 249	1 – 249	250+
United States	1 – 9	10 – 99	100 – 499	1 – 449	500+

Fonte: OECD (2010), adaptado de Berisha e Pula (2015).

No contexto brasileiro, o Banco Nacional de Desenvolvimento Econômico e Social (BNDES) define o porte empresarial com base na Receita Operacional Bruta Anual, conforme exposto na Tabela 4.

Tabela 4: Classificação de empresas no Brasil segundo a Receita Operacional Bruta Anual.

Classificação	Receita operacional bruta anual ou renda anual
Microempresa	Menor ou igual a R$ 360 mil
Pequena empresa	Maior que R$ 360 mil e menor ou igual a R$ 4,8 milhões
Média empresa I	Maior que R$ 4,8 milhões e menor ou igual a R$ 90 milhões
Média empresa II	Maior que R$ 90 milhões e menor ou igual a R$ 300 milhões
Grande empresa	Maior que R$ 300 milhões

Fonte:Adaptado de BNDES (2022).

Essas diferentes formas de definição reforçam a importância de compreender as particularidades estruturais e econômicas das PMEs ao desenvolver boas práticas e guias gerais, garantindo que as recomendações possam ser aplicáveis em diferentes realidades (BERISHA; PULA, 2015).

4.2 Conceitos e fundamentos de ETL e pipelines de dados

O termo pipeline de dados refere-se ao conjunto organizado de processos responsáveis por mover, transformar e disponibilizar dados desde a sua origem até um destino final, de forma contínua. Ele abrange desde a coleta de dados em fontes diversas até a entrega dos

dados tratados e prontos para análise (MEHMOOD; ANEES, 2020). Embora pipelines de dados possam incluir integrações em tempo real ou por lotes, neste trabalho o foco principal recai sobre os pipelines baseados em ETL, por serem mais aderentes à realidade de PMEs.

O processo de ETL consolidou-se como um dos pilares fundamentais na arquitetura de sistemas analíticos modernos, desempenhando papel central na preparação e integração de dados provenientes de múltiplas fontes. O ETL surgiu para reunir dados que estavam espalhados em diferentes lugares, o que permite fazer análises centralizadas e ajuda nas decisões importantes da empresa. O processo de ETL tem três etapas principais que funcionam juntas de forma contínua.

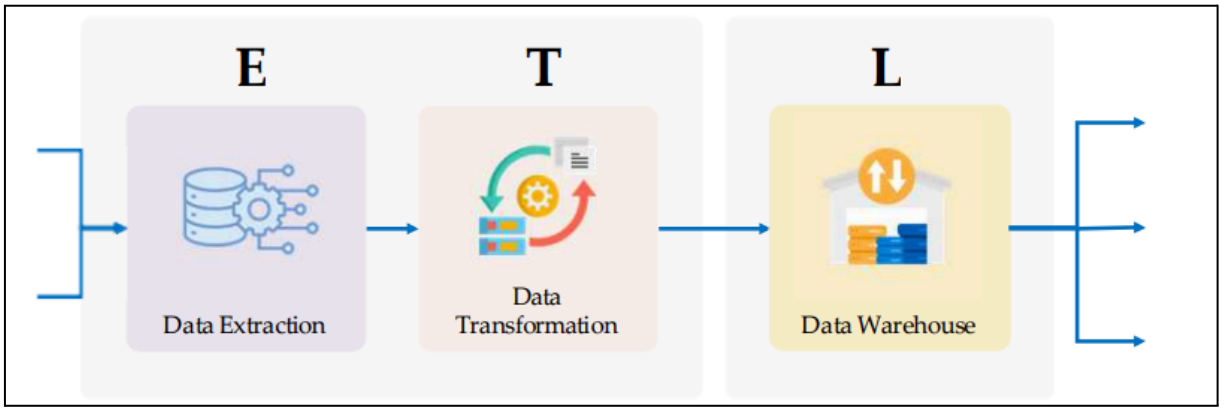
A Extração consiste na coleta de dados de diversas fontes, tais como bancos relacionais, APIs, arquivos CSV, logs, sistemas legados, etc. Essa etapa requer definição clara de quais dados serão capturados e atenção especial à consistência e à integridade das fontes, prevenindo problemas que podem aparecer nas fases seguintes. Uma extração bem planejada reduz custos operacionais e aumenta a qualidade geral do pipeline (MEHMOOD; ANEES, 2020). Além disso, é comum utilizar estratégias como snapshot, incremental load ou micro-batch controlado para otimizar a performance e minimizar impactos nas fontes originais (NWOKEJI; MATOVU, 2021).

A Transformação é a fase em que os dados brutos são refinados para se tornarem consistentes e úteis para análise. São realizadas atividades como limpeza de valores nulos, remoção de duplicidades, normalização de formatos (por exemplo, datas e moedas) e cálculos derivados, além de junções para enriquecimento contextual. A rastreabilidade e o registro detalhado de cada transformação são fundamentais para garantir confiabilidade e possibilitar auditorias futuras (RAGHAV et al., 2024; BILAL KHAN et al., 2024). Em PMEs, priorizar transformações automáticas e progressivas ajuda a reduzir custos e a dependência de revisões manuais (ALSIBHAWI et al., 2023).

A Carga consiste na inserção dos dados tratados no ambiente de destino, que pode ser um data warehouse, um data lake ou um banco relacional especializado em análises. Nessa etapa, definem-se políticas de atualização, como inserção incremental ou substituição completa, e são implementados mecanismos para garantir integridade, backup e versionamento. Uma carga bem realizada assegura que os dados finais estejam prontos para alimentar dashboards e análises de negócio, oferecendo confiabilidade para a tomada de decisão (NWOKEJI; MATOVU, 2021).

Para ilustrar esse fluxo de forma clara e intuitiva, a Figura 1 apresenta uma visão geral das etapas ETL, destacando a sequência de extração, transformação e carga, conforme proposto por Bilal Khan et al. (2024).

Figura 1: Arquitetura geral do processo ETL.



Fonte: Bilal Khan et al. (2024).

Outro conceito central na evolução dos pipelines é a modularidade, que garante a separação das etapas em unidades independentes e integráveis. Essa abordagem favorece manutenção facilitada, evolução incremental e maior controle sobre o fluxo de dados (ALI; WREMBEL, 2017). Ao modularizar, cada componente pode ser monitorado, testado e ajustado isoladamente, aumentando a confiabilidade geral do pipeline (CHIKHALKAR et al., 2025).

Com o amadurecimento das demandas analíticas, surgiram variações como o modelo ELT (Extract, Load, Transform), no qual os dados são primeiro carregados no repositório e só depois transformados, aproveitando o poder computacional dos data warehouses modernos. Em paralelo, abordagens como o micro-batch processing, que processa dados em pequenos lotes com intervalos curtos, surgem como solução intermediária, oferecendo maior controle operacional e custos reduzidos em comparação ao streaming puro, sendo uma opção bastante viável para PMEs (DAPKUTE et al., 2024; JOHNSON et al., 2024).

Para pequenas e médias empresas, entender esses conceitos é ainda mais estratégico, visto que simplicidade, escalabilidade e adaptabilidade são essenciais para viabilizar iniciativas baseadas em dados. A clareza na estruturação de pipelines de dados pode ser o diferencial necessário para transformar grandes volumes de dados brutos em insights valiosos, mesmo em cenários com recursos financeiros e técnicos limitados. Ao compreender detalhadamente esses fundamentos, as PMEs estarão mais preparadas para amadurecer suas

iniciativas de dados, adotar melhores práticas e aumentar sua competitividade em um cenário de negócios cada vez mais orientado a dados.

4.3 Arquitetura técnica e modelo dimensional em pipelines ETL

A arquitetura de um pipeline ETL é um dos elementos centrais para assegurar que o fluxo de dados em uma organização seja eficiente, seguro e capaz de sustentar análises estratégicas. A construção dessa arquitetura envolve uma série de componentes que trabalham juntos para coletar, transformar e disponibilizar dados de maneira confiável. Para PMEs, compreender e adotar uma arquitetura bem planejada pode representar um diferencial competitivo fundamental (JOHNSON et al., 2024).

As fontes de dados são o ponto inicial de qualquer pipeline ETL. A diversidade e heterogeneidade dessas fontes requerem um planejamento cuidadoso para garantir que os dados sejam coletados corretamente e se mantenham íntegros. A capacidade de lidar com múltiplos formatos e fluxos de dados é um dos principais desafios em ambientes empresariais dinâmicos (MEHMOOD; ANEES, 2020).

Após a extração, os dados são encaminhados para a staging area, também chamada de área de preparo. Essa camada funciona como um buffer temporário onde os dados são armazenados antes das transformações mais complexas. A staging area não é exclusiva de um modelo específico, podendo ser utilizada em arquiteturas ETL ou ELT. Seu principal papel é isolar as fontes originais de possíveis impactos durante as transformações, facilitar o versionamento e permitir auditorias detalhadas (ALI; WREMBEL, 2017).

Na sequência, os dados passam pelos motores de transformação, responsáveis por torná-los utilizáveis e confiáveis. As operações típicas incluem:

- Limpeza: remoção de valores nulos ou inconsistentes, correção de outliers e padronização de formatos.
- Padronização: unificação de unidades de medida, datas e nomenclaturas.
- Enriquecimento: junção com outras fontes, como dados demográficos ou de mercado, para acrescentar contexto.
- Agregações e cálculos derivados: criação de métricas, médias ou segmentações específicas.

Tecnologias como scripts em Python, frameworks como Spark ou dbt e rotinas SQL são amplamente utilizadas nessa etapa. A transformação visa garantir consistência e preparar os dados para análises estratégicas (DINESH; DEVI, 2024).

Após a transformação, os dados são carregados em um Data Warehouse (DW) ou em Data Lakes, dependendo do modelo de armazenamento adotado. O DW oferece visão estruturada e consolidada, facilitando análises gerenciais e relatórios. Já os data lakes são indicados para armazenar grandes volumes de dados não estruturados ou semiestruturados. Para PMEs, o uso de um DW pode ser mais viável, pois oferece simplicidade de consulta e maior controle de governança (ZARATE et al., 2024).

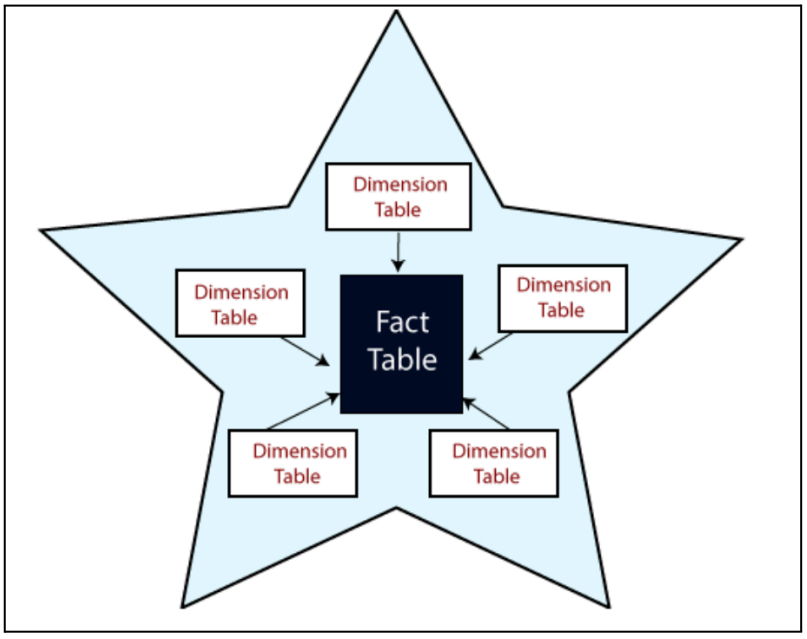
Dentro do DW, a modelagem dimensional assume papel central. Essa modelagem organiza os dados em tabelas fato e tabelas dimensão:

- Tabelas fato concentram dados quantitativos e métricas, como vendas ou receita, e contêm chaves estrangeiras que ligam às dimensões.
- Tabelas dimensão armazenam informações descritivas que contextualizam os fatos, como nomes de clientes, produtos ou datas.

O modelo estrela (star schema) é o mais utilizado, com a tabela fato centralizada e dimensões conectadas diretamente. Essa estrutura facilita consultas rápidas e reduz a necessidade de junções complexas. Além disso, promove clareza para equipes com menor experiência técnica, sendo especialmente útil para PMEs (WIJAYA; WIRATAMA; WIJAYA, 2024).

Na prática, a tabela fato centraliza dados como quantidade vendida ou receita total, enquanto as dimensões contêm atributos detalhados, como período (ano, mês), produto (SKU, categoria), cliente (segmento) ou canal de venda. A Figura 2 ilustra essa configuração.

Figura 2: Modelo Star Schema em Data Warehouse.



Fonte: Wijaya; Wiratama; Wijaya (2024).

Ao adotar o modelo estrela, as PMEs conseguem integrar informações de diversas áreas em uma única base, fortalecendo o alinhamento estratégico e apoiando decisões baseadas em dados.

Também é comum o uso de data marts, subconjuntos do DW dedicados a áreas específicas (como vendas ou marketing). Essa segmentação permite flexibilidade analítica sem comprometer o repositório central.

Após a carga, os dados passam a alimentar dashboards e ferramentas de visualização, funcionando como a interface final entre o pipeline técnico e os usuários de negócio. Embora esta pesquisa não aprofunde a camada de BI, é essencial que o pipeline forneça dados com clareza, precisão e rastreabilidade para suportar análises (JOHNSON et al., 2024).

No que se refere aos padrões de projeto, destacam-se:

- Modularidade: divisão clara das etapas (extração, transformação, carga), permitindo reuso e manutenção facilitada. A modularidade também contribui para maior robustez em caso de falhas em uma etapa específica.
- Orquestração: organização e controle do fluxo de execução, garantindo que as tarefas ocorram em ordem correta e com dependências gerenciadas.
- Monitoramento e rastreabilidade: uso de logs detalhados, versionamento e auditorias contínuas para assegurar qualidade, segurança e transparência.

Por fim, conceitos de flexibilidade e escalabilidade são indispensáveis. A flexibilidade permite rápida adaptação a novas fontes ou requisitos, enquanto a escalabilidade garante funcionamento contínuo mesmo com aumento de volume. Estratégias como particionamento horizontal, clusters distribuídos e paralelismo são fundamentais para alcançar esses objetivos (MEHMOOD; ANEES, 2020).

Uma arquitetura bem planejada fortalece a governança, melhora a eficiência operacional e alinha as operações de dados com os objetivos estratégicos. Para PMEs, a clareza na definição de cada componente e a adoção de boas práticas podem democratizar o acesso a análises avançadas, fomentar a inovação e aumentar a competitividade.

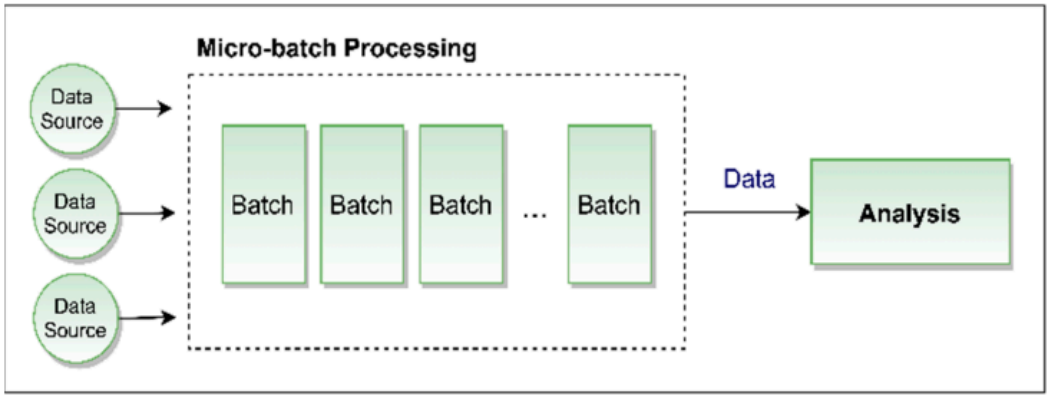
4.3.1 Micro-Batch Processing vs Stream Processing

A evolução dos pipelines de dados trouxe consigo diferentes abordagens para o processamento de informações, com destaque para o Micro-Batch Processing e o Stream

Processing. Compreender as diferenças entre essas abordagens é essencial para definir estratégias de integração e análise de dados viáveis.

O Micro-Batch Processing consiste no processamento de dados em pequenos lotes (mini-batches), agrupados em intervalos curtos e processados periodicamente. Essa técnica busca aproximar-se de uma experiência de tempo quase real, mas sem a complexidade total do streaming puro. Em PMEs, essa abordagem é particularmente vantajosa por equilibrar desempenho e simplicidade técnica, além de apresentar custos de manutenção mais baixos e maior resiliência a falhas (DAPKUTE et al., 2024; JOHNSON et al., 2024). A principal limitação está na latência: mesmo com lotes pequenos e execuções frequentes, como a cada minuto, existe um pequeno atraso comparado ao processamento contínuo. Entretanto, para muitos cenários em PMEs, essa latência é considerada aceitável, pois permite conciliar agilidade analítica com sustentabilidade operacional (SHANMUKHA, 2022). Ferramentas como Apache Spark Structured Streaming e fluxos agendados no Airflow são amplamente utilizadas para operacionalizar micro-batches, permitindo execuções escaláveis e previsíveis. A Figura 3 ilustra o fluxo geral do Micro-Batch Processing, destacando a divisão em pequenos lotes antes da etapa de análise final.

Figura 3: Representação geral do Micro-Batch Processing.

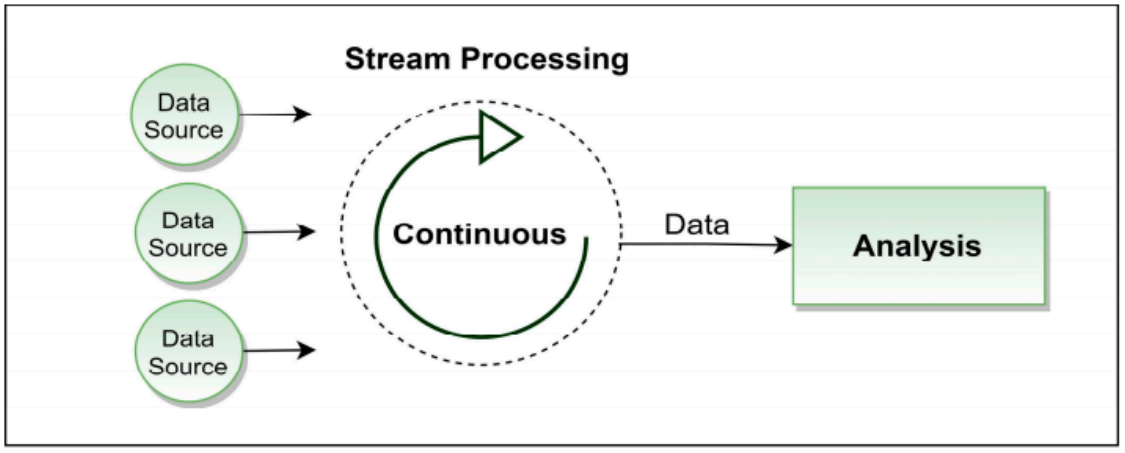


Fonte: Dapkute et al. (2024).

O Stream Processing, por sua vez, se baseia no processamento contínuo de cada evento individual assim que ele chega ao sistema, sem formação de lotes. Essa abordagem é a mais próxima do real-time e é essencial em contextos que exigem latência mínima, como monitoramento de sensores, detecção de fraudes financeiras ou sistemas de recomendação instantânea (DAPKUTE et al., 2024). Entre suas principais vantagens está a capacidade de responder imediatamente a eventos críticos, garantindo atualização quase instantânea das informações. No entanto, sua adoção impõe desafios significativos, como maior complexidade técnica, exigência de maior capacidade computacional e custos operacionais

elevados (BILAL KHAN et al., 2024). Para PMEs, o Stream Processing pode ser inviável em fases iniciais, pois demanda equipes mais especializadas e investimentos robustos em infraestrutura. A Figura 4 demonstra o fluxo de dados contínuo característico dessa abordagem, evidenciando a atualização imediata das análises.

Figura 4: Representação geral do Stream Processing.



Fonte: Dapkute et al. (2024).

A escolha entre as duas técnicas depende de fatores como latência exigida, capacidade técnica, orçamento disponível e objetivos de negócio. Em termos de latência, o Micro-Batch Processing oferece desempenho satisfatório com pequenos atrasos (em segundos ou minutos), o que costuma ser suficiente para relatórios frequentes e dashboards atualizados quase em tempo real. Já o Stream Processing trabalha com latência mínima, ideal para aplicações críticas que exigem resposta imediata.

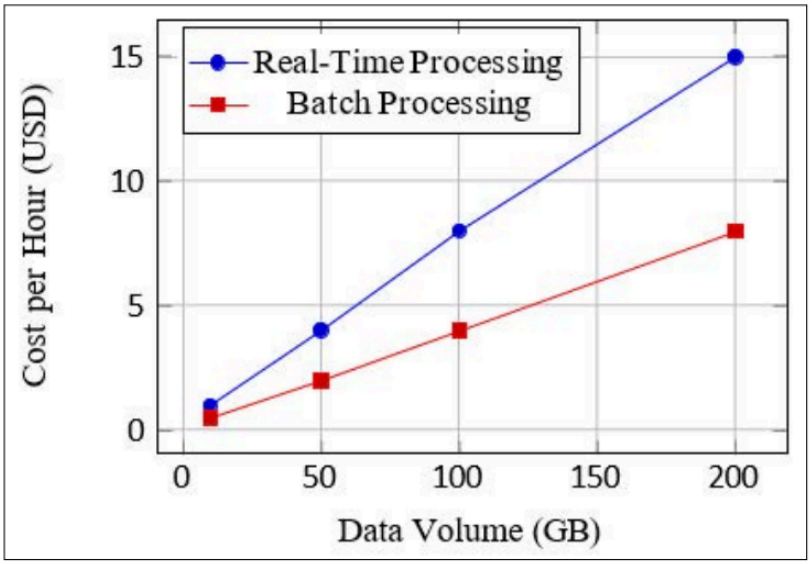
Do ponto de vista da complexidade, o Micro-Batch é considerado moderado, exigindo menos ajustes em infraestrutura e podendo ser operacionalizado pelo uso de ferramentas. Por outro lado, o Stream Processing demanda arquiteturas robustas, maior monitoramento e constante ajuste de performance, o que eleva significativamente o nível de complexidade técnica.

A viabilidade para PMEs é, portanto, um critério decisivo. O Micro-Batch tende a ser mais adequado, equilibrando agilidade e investimento moderado. Já o Stream Processing pode ser considerado apenas em situações específicas nas quais o tempo real é absolutamente necessário e há orçamento compatível com a complexidade da solução.

Além desses aspectos, é importante considerar o impacto financeiro em função do volume de dados processados. A Figura 5 ilustra essa diferença de custos, mostrando que o

processamento em tempo real torna-se exponencialmente mais caro à medida que o volume aumenta (MANDALA, 2023).

Figura 5: Custo por hora em função do volume de dados: real-time vs. batch processing.



Fonte: Mandala (2023).

4.4 Critérios para seleção de ferramentas e tecnologias ETL

A escolha adequada de ferramentas e tecnologias para pipelines ETL é um fator determinante para o sucesso de estratégias de dados em PMEs. Devido a restrições orçamentárias, limitações de infraestrutura e necessidade de flexibilidade, essa decisão se torna ainda mais crítica nesse tipo de organização. Segundo Bilal Khan et al. (2024), a seleção tecnológica impacta diretamente a escalabilidade, o desempenho e a sustentabilidade operacional, sendo fundamental optar por soluções compatíveis, modulares e de fácil manutenção.

Do ponto de vista técnico, é essencial adotar ferramentas que permitam reuso de componentes, integração com múltiplas fontes e suporte a scripts personalizados. A flexibilidade na integração com diferentes formatos de dados também é um critério chave, especialmente quando combinada com a capacidade de armazenamento em nuvem.

A adoção de soluções em nuvem, como AWS, Google Cloud ou Azure, tende a ser mais vantajosa para PMEs do que estruturas on-premise, por eliminar grandes investimentos iniciais em hardware e permitir escalabilidade sob demanda. Em termos de custo, o armazenamento em nuvem pode variar entre US$ 0,023 e US$ 0,25 por GB/mês, além de taxas por transferência de dados (MANDALA, 2023).

Neste trabalho, a recomendação pelo Amazon S3 como repositório principal destaca-se por sua alta durabilidade, escalabilidade elástica e segurança avançada (MANDALA, 2023). Além disso, possui o modelo de cobrança variável (pay-as-you-go), o que reduz barreiras de entrada para PMEs. Sua arquitetura baseada em buckets permite versionamento, replicação geográfica e criptografia automática. Outro ponto positivo é a compatibilidade com ferramentas analíticas e de visualização, como o Amazon QuickSight.

Quanto aos custos, ferramentas open-source como o Apache Airflow ganham destaque. Além de serem livres de licenciamento, oferecem alto grau de flexibilidade e uma comunidade ativa de suporte. Em comparação, plataformas comerciais como Talend ou Informatica podem ultrapassar US$ 100 mil/ano em custos (DAPKUTE et al., 2024).

A decisão pelo Airflow como ferramenta de orquestração neste trabalho justifica-se por sua arquitetura robusta baseada em DAGs (Directed Acyclic Graphs), suporte a execução paralela, tolerância a falhas e compatibilidade com ferramentas de monitoramento como Grafana e Prometheus. Segundo Yasmin et al. (2025), 85% dos usuários relataram ganhos em rastreabilidade e visibilidade dos pipelines após sua adoção.

A Tabela 5 apresenta uma comparação entre frameworks de orquestração amplamente utilizados, com destaque para o Airflow, reforçando a coerência da escolha adotada neste trabalho.

Tabela 5: Comparação entre frameworks de orquestração de pipelines.

Feature	**NiFi**	**Airflow**	**Mage**
Data Flow	Visual flow design	Workflow DAG	Workflow DAG
Language	Java	Python	Java
Real-time Processing	Yes	No	Yes
Batch Processing	Yes	Yes	Yes
Scalability	High	High	High
Community Support	Large	Large	Growing
Extensibility	Processor API	Plugins	Plugin API
Monitoring	Built-in	External tooling	Built-in
Fault Tolerance	Yes	Yes	Yes
Workflow Scheduling	Limited	Yes	Yes
Analytic Oriented	Yes	Yes	Yes
Publication Functionalities	Yes	Yes	Yes
Standards	Compliance support	Compliance support	Compliance support

Fonte: Zarate et al. (2024).

Nas etapas de extração e transformação, o uso do Python se mostra vantajoso por sua vasta biblioteca de recursos (como pandas e SQLAlchemy), sua eficiência no tratamento de

grandes volumes de dados e sua baixa curva de aprendizado, que é uma característica relevante para equipes pequenas (MEHMOOD; ANEES, 2020).

Dessa forma, a combinação entre Airflow (orquestração), Python (extração e transformação) e AWS S3 (armazenamento) representa uma solução técnica robusta, modular e economicamente viável, perfeitamente alinhada às necessidades e limitações típicas de PMEs. Essa arquitetura orienta a proposta prática deste trabalho, que visa entregar um modelo replicável e sustentável para ambientes empresariais de menor porte.

4.5 Especificidades e desafios enfrentados por PMEs na adoção de pipelines de dados

A adoção de pipelines de dados e processos ETL por PMEs representa um movimento estratégico fundamental para modernizar o uso da informação. No entanto, essa transição é marcada por desafios diversos que envolvem desde aspectos técnicos até questões gerenciais e financeiras.

Entre os principais entraves técnicos está a carência de infraestrutura robusta, que compreende servidores, ferramentas de orquestração e soluções de armazenamento em nuvem. Johnson et al. (2024) apontam que muitas PMEs não dispõem de recursos para manter ambientes escaláveis e tolerantes a falhas. Outro obstáculo crítico é a heterogeneidade dos dados, pois essas empresas frequentemente operam com fontes variadas (sistemas legados, bancos relacionais, planilhas e APIs) exigindo adaptações para lidar com estruturas distintas. Essa variedade, aliada à falta de padronização, aumenta significativamente a complexidade e o custo de integração (MEHMOOD; ANEES, 2020).

De acordo com Nwokeji e Matovu (2021), os principais desafios enfrentados em processos ETL incluem: aumento da complexidade dos dados, altos custos operacionais, consumo excessivo de tempo, dificuldades de manutenção diante de mudanças, ausência de automação e dependência de processos manuais, além da falta de padrões consolidados. Tais dificuldades prejudicam a escalabilidade e sustentabilidade dos fluxos de dados, mesmo quando as soluções iniciais são implementadas com sucesso.

No campo gerencial, destaca-se a ausência de uma cultura orientada a dados. Becker et al. (2023) relatam que há forte resistência à adoção tecnológica em muitas PMEs brasileiras, influenciada por receios de ruptura em processos consolidados. A escassez de profissionais especializados enfatiza a situação: Alsibhawi et al. (2023) observam que a falta de expertise interna compromete a implementação e manutenção eficaz de pipelines. Além disso, a

inexistência de políticas de governança bem definidas resulta em iniciativas fragmentadas, com acúmulo de dados redundantes e baixa rastreabilidade estratégica.

Do ponto de vista financeiro, as limitações orçamentárias são especialmente restritivas. Segundo Becker et al. (2023), muitas PMEs operam com margens reduzidas, o que dificulta o investimento em tecnologias avançadas. Embora frameworks open-source, como Apache Airflow, representem alternativas viáveis para redução de custos, essas soluções ainda exigem profissionais qualificados para instalação e manutenção, realimentando o ciclo de barreiras técnicas e gerenciais. Soma-se a isso a dificuldade em mensurar o retorno sobre o investimento (ROI), o que gera insegurança e retarda decisões de adoção tecnológica.

Para mitigar esses desafios, a adoção gradual de soluções se mostra uma estratégia promissora. Zarate et al. (2024) defendem que a implementação progressiva, por meio de provas de conceito, permite testar abordagens de forma controlada, reduzir riscos e realizar ajustes antes da escala total. Raghav et al. (2024) complementam ao sugerir a modularização dos componentes técnicos, permitindo maior controle, menor custo inicial e melhor adaptação das equipes. Por fim, o investimento contínuo em capacitação é apontado como elemento essencial para garantir a sustentabilidade das iniciativas analíticas e consolidar uma cultura organizacional baseada em dados.

4.6 Benefícios estratégicos do uso de ETL em pequenas e médias empresas

A adoção dos processos ETL transcende o aspecto técnico, configurando-se como um diferencial estratégico. Ao consolidar informações e automatizar fluxos operacionais, o ETL permite decisões mais ágeis e embasadas, promovendo ganhos em competitividade, eficiência e inovação (BECKER et al., 2023).

Com a integração eficaz de dados, as PMEs passam a contar com análises preditivas mais precisas e indicadores operacionais em tempo quase real. Isso viabiliza a personalização de serviços e a adaptação rápida a demandas de mercado, aspectos valorizados no ambiente atual (TAWIL et al., 2024). Os pipelines também abrem caminho para a criação de data products, capazes de gerar novas receitas e diferenciar a atuação da empresa. Ao centralizar dados de diversas fontes, tornam-se possíveis o mapeamento de nichos, a antecipação de tendências e a aceleração da transformação digital (ZARATE et al., 2024).

Do ponto de vista operacional, o ETL reduz tarefas repetitivas e erros humanos, aumentando a confiabilidade da informação e liberando a equipe para tarefas estratégicas. A automação contribui ainda para a redução de custos e melhora o uso dos recursos internos

Pipelines bem projetados também simplificam a manutenção futura e reduzem riscos em atualizações, graças à uniformização estrutural dos dados e ao uso de versionamento contínuo. Além disso, quando estruturados com mecanismos de rastreamento e controle, possibilitam maior visibilidade sobre o retorno sobre investimento (ROI), reforçando a tomada de decisão baseada em evidências (JOHNSON et al., 2024).

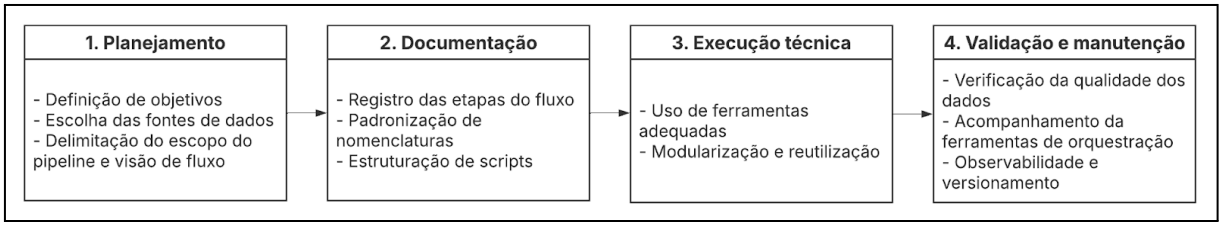
Por fim, a governança de dados embutida nesses processos fortalece a conformidade com normas e aumenta a confiança de clientes, parceiros e investidores. Assim, a implantação de ETL deve ser vista como uma estratégia fundamental para o crescimento sustentável das PMEs no cenário digital contemporâneo.

4.7 Boas práticas para construção de pipelines ETL em PMEs

A adoção de pipelines de dados em PMEs exige a aplicação de boas práticas que conciliem rigor técnico, simplicidade operacional e viabilidade econômica. Diante de restrições comuns nesses contextos, como escassez de recursos, infraestrutura limitada e baixa maturidade analítica, torna-se fundamental adotar diretrizes adaptadas que aumentem a efetividade e a sustentabilidade das iniciativas de dados (CHIKHALKAR et al., 2025; ALSIBHAWI et al., 2023).

Para contextualizar esse conjunto de práticas, a Figura 6 a seguir apresenta uma visão geral dos principais pilares envolvidos na construção de pipelines de dados em PMEs, construída a partir da análise dos dezenove artigos da pesquisa bibliográfica:

Figura 6: Etapas de boas práticas para construção de pipelines.



Fonte: Elaborado pelo autor.

4.7.1 Aspectos técnicos recomendados

A modularidade é uma das práticas mais enfatizadas na literatura. A separação das etapas de extração, transformação e carga permite que cada componente seja desenvolvido, testado e mantido de forma independente, facilitando reuso, escalabilidade progressiva e diagnóstico de falhas (CHIKHALKAR et al., 2025). Esse princípio favorece ambientes com infraestrutura restrita, pois reduz riscos e melhora a flexibilidade operacional.

A automação por orquestração, com uso de ferramentas como o Apache Airflow, permite programar execuções periódicas, controlar dependências entre tarefas e monitorar falhas. No Airflow, os fluxos são organizados em estruturas conhecidas como DAGs (Directed Acyclic Graphs), que representam visualmente o encadeamento de tarefas. Essa abordagem torna o pipeline mais rastreável, documentado e resiliente a falhas.

A rastreabilidade (observabilidade), associada a mecanismos robustos de logging, é indispensável para assegurar a integridade dos dados e o controle de versões. Implementar logs detalhados em todas as etapas facilita a auditoria e reduz o tempo de resposta em caso de inconsistências. A rastreabilidade completa dos dados, desde sua origem até os dashboards, é um requisito essencial de governança em ambientes orientados a dados (ROZONY et al., 2024).

A validação contínua dos dados, por meio de testes automatizados (unitários e lógicos), é outra prática recomendada. Devem ser implementadas verificações de consistência, como checagem de tipos, conformidade com regras de negócio e integridade referencial (MEHMOOD; ANEES, 2020). Ferramentas como Great Expectations ou validadores nativos no Airflow podem ser utilizados para esse fim, mesmo com equipes reduzidas.

4.7.2 Estrutura recomendada do pipeline ETL

Com base na pesquisa bibliográfica, propõe-se um modelo estruturado de pipeline que pode ser utilizado como referência por PMEs interessadas em adotar processos ETL de forma gradual e segura. Esse fluxo incremental não representa necessariamente a PoC aplicada neste trabalho, mas constitui uma recomendação teórica fundamentada nos estudos analisados.

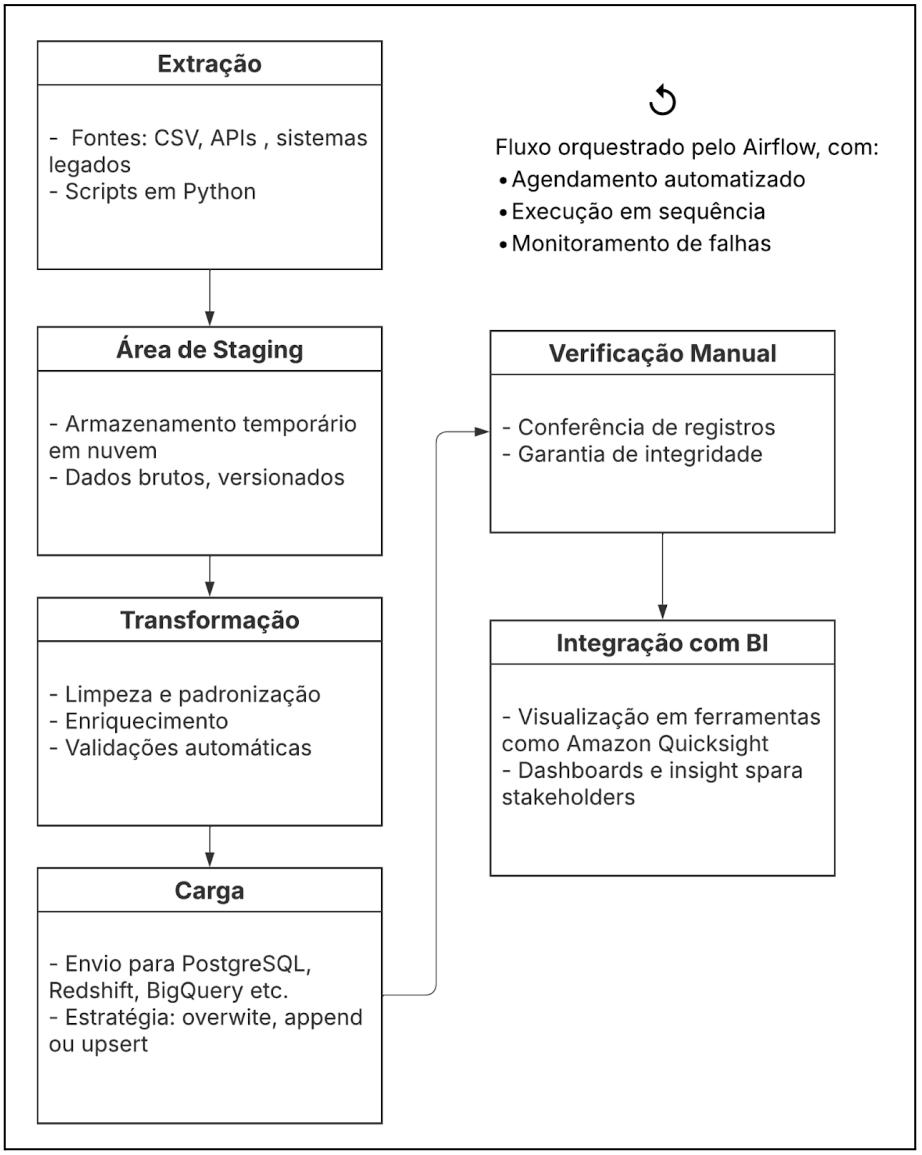
O pipeline recomendado segue as etapas clássicas:

- Extração: A coleta dos dados pode envolver múltiplas fontes, como sistemas legados, arquivos CSV, APIs externas, etc. É essencial realizar testes prévios de conectividade, autenticação e verificação de schema, utilizando scripts em Python com bibliotecas como requests, pandas e SQLAlchemy.
- Staging Area: Uma área intermediária, geralmente em nuvem (por exemplo, AWS S3), armazena temporariamente os dados antes da transformação. Isso evita impactos diretos nas fontes de dados e permite versionamento e reprocessamento seguro.
- Transformação: Consiste em limpar, padronizar, enriquecer e validar os dados coletados. A padronização semântica e o enriquecimento com fontes auxiliares (ex.: dados geográficos ou categorização de clientes) aumentam a utilidade analítica.

- Carga: Os dados transformados são enviados para repositórios como data warehouses (Redshift, BigQuery), bancos relacionais (PostgreSQL, MySQL) ou armazenamentos em nuvem. Estratégias de overwrite, append ou upsert devem ser adotadas conforme o tipo e o volume dos dados. O uso de ferramentas como o DBeaver permite verificação pós-carga e testes manuais.
- Verificação Manual: Um ponto de controle adicional pode ser implementado para auditoria e conferência de indicadores críticos, especialmente em ambientes de baixa automação.
- Integração com BI: A última etapa consiste em disponibilizar os dados para ferramentas de visualização como o Amazon QuickSight, promovendo insights em tempo real e integração com stakeholders.

  A Figura 7 a seguir resume visualmente o fluxo recomendado de um pipeline ETL para PMEs, construído a partir da pesquisa bibliográfica:

Figura 7: Fluxograma geral do pipeline ETL proposto.



Fonte: Elaborado pelo autor.

4.7.3 Considerações gerenciais, perfil profissional e governança

A implementação de boas práticas técnicas deve ser acompanhada por ações gerenciais estruturadas, como a definição de políticas de governança de dados. Isso inclui atribuição de donos pelos conjuntos de dados, definição de níveis de acesso e permissionamento, padronização de nomenclaturas, versionamento por Git e documentação contínua (CHIKHALKAR et al., 2025).

Além das estruturas de gestão, a qualificação do time técnico é essencial. O profissional responsável pela construção dos pipelines deve possuir conhecimentos em engenharia de dados, programação em Python, arquitetura de pipelines e governança de dados. Também são recomendáveis competências em bancos de dados (SQL e NoSQL),

versionamento de código (Git) e ferramentas de monitoramento. A formação pode incluir cursos em ciência da computação, sistemas de informação, engenharia de software ou áreas afins, com foco em experiência prática em projetos de dados.

Do ponto de vista organizacional, a promoção de uma cultura orientada a dados é igualmente relevante. Programas de treinamento, workshops e ações de sensibilização facilitam a aceitação das soluções e aumentam o engajamento interno.

Por fim, a adoção incremental, iniciando com fluxos simples validados por PoCs, permite a evolução gradual da solução com menor risco e maior aderência à realidade da empresa. A modularização técnica e a capacitação progressiva fortalecem a sustentabilidade do projeto a longo prazo (DINESH; DEVI, 2024).

5. METODOLOGIA

Esta seção apresenta o percurso metodológico adotado para o desenvolvimento do estudo, descrevendo as etapas e estratégias utilizadas para atingir os objetivos propostos. A pesquisa foi estruturada com base em duas frentes complementares: a Pesquisa Bibliográfica, responsável pela fundamentação teórica e identificação de boas práticas, e a PoC, voltada à validação prática das diretrizes em um ambiente simulado, representativo do contexto de PMEs.

5.1 Abordagem metodológica

A abordagem adotada é qualitativa, exploratória e descritiva, com o objetivo de compreender, estruturar e aplicar boas práticas para a construção de pipelines de dados em PMEs. Essa combinação metodológica justifica-se pela natureza emergente do tema e pela escassez de estudos consolidados na área. Conforme Gil (2019), pesquisas exploratórias são recomendadas quando o objeto investigado ainda está em fase de amadurecimento, permitindo o levantamento de hipóteses e direcionamentos iniciais.

5.2 Pesquisa bibliográfica

A pesquisa bibliográfica foi conduzida com o objetivo de identificar práticas recomendadas, ferramentas acessíveis e desafios recorrentes na implementação de processos ETL em pequenas e médias empresas. Foram consultadas bases como Google Scholar e IEEE Xplore, considerando inicialmente publicações entre 2020 e 2025. Uma publicação de 2017 foi adicionada para ajudar no embasamento teórico. Os critérios de seleção priorizaram a

relevância conceitual e a aplicabilidade técnica das abordagens, mesmo em estudos com escopo mais amplo, como integração de dados em ambientes corporativos, pipelines analíticos em nuvem, ferramentas open-source e estratégias de transformação digital.

Embora o foco estivesse em PMEs, a escassez de estudos específicos exigiu uma abordagem qualitativa e contextualizada da literatura. Foram selecionados trabalhos que, mesmo que ainda não tratassem exclusivamente do setor de pequenas empresas, apresentavam diretrizes e soluções aplicáveis a esse contexto, especialmente em termos de modularidade, automação e uso de tecnologias de baixo custo.

A Tabela 6 apresenta os dezenove trabalhos utilizados como referência para a sistematização das boas práticas desenvolvidas neste estudo. Os artigos estão organizados por título, autores e ano de publicação.

Tabela 6: Trabalhos utilizados como base para a definição das boas práticas de ETL em PMEs.

Título	Autores	Ano de Publicação
Developing scalable data solutions for small and medium enterprises: challenges and best practices	JOHNSON, Ebunoluwa et al.	2024
A data pipeline concept for digitizing services in small and medium-sized companies	CHIKHALKAR, Akshay et al.	2025
Business intelligence adoption for small and medium enterprises: conceptual framework	ALSIBHAWI, Ibrahim Abdusalam Abubaker; YAHAYA, Jamaiah Binti; MOHAMED, Hazura Binti	2023
Trends and challenges towards an effective data-driven decision making in UK SMEs: case studies and lessons learnt from the analysis of 85 SMEs	TAWIL, Abdel-Rahman H. et al.	2024
Digital twin data management: framework and performance metrics of cloud-based ETL system	DAPKUTE, Austeja et al.	2024
An efficient hybrid optimization of ETL process in data warehouse of cloud architecture	DINESH, Lina; DEVI, K. Gayathri	2024
ETL: from design to deployment	RAGHAV; BARANI; VIJAY RAM	2024
An overview of ETL techniques, tools, processes and evaluations	KHAN, Bilal et al.	2024

in data warehousing		
Efficient ETL processes: a comparative study of Apache Airflow vs. traditional methods	EETI, Shanmukha	2022
Evolution of extract-transform-load (ETL) processes towards data product pipelines	ZARATE, Gorka et al.	2024
An efficient hybrid optimization of ETL process in data warehouse of cloud architecture	MEHMOOD, Erum; ANEES, Tayyaba	2020
A systematic literature review on big data extraction, transformation and loading (ETL)	NWOKEJI, Joshua C.; MATOVU, Richard	2021
A systematic review of big data integration challenges and solutions for heterogeneous data sources	ROZONY, Farhana Zaman et al.	2024
From conceptual design to performance optimization of ETL workflows: current state of research and open problems	ALI, Syed Muhammad Fawad; WREMBEL, Robert	2017
Big data em micro e pequenas empresas: uma revisão sistemática	BECKER, Michel et al.	2023
An empirical study of developers' challenges in implementing Workflows as Code: a case study on Apache Airflow	YASMIN, Jerin et al.	2025
Data quality management and performance optimization for enterprise-scale ETL pipelines in modern analytical ecosystems	MACHIREDDY, Jeshwanth Reddy	2023
Implementation of data warehouse and star schema for optimizing property business decision making	WIJAYA, Willsen et al.	2024
Advances in distributed storage systems for big data	MANDALA, Nishanth Reddy	2023

5.3 Etapa prática e validação

A segunda etapa da pesquisa consistiu na elaboração de uma prova de conceito com o objetivo de validar, em ambiente controlado, as boas práticas identificadas na literatura. A PoC foi concebida com foco em viabilidade técnica, modularidade, automação e governança, simulando condições comuns em PMEs.

A solução prática foi construída utilizando ferramentas amplamente reconhecidas por sua flexibilidade e custo reduzido, como Apache Airflow, Docker, PostgreSQL e DBeaver. A seleção tecnológica buscou refletir um cenário realista e replicável para empresas com recursos limitados, permitindo avaliar o impacto das boas práticas de forma estruturada e aplicada.

6. RESULTADOS E DISCUSSÃO

Esta seção apresenta os principais resultados obtidos a partir da aplicação prática desenvolvida neste estudo. O objetivo foi demonstrar a viabilidade técnica da construção de um pipeline de dados moderno, modular e orquestrado, utilizando ferramentas acessíveis e adaptáveis ao contexto de PMEs. A PoC foi executada em ambiente simulado, com base em dados reais disponibilizados publicamente, a fim de validar o fluxo completo de extração, transformação, carga e orquestração automatizada.

O desenvolvimento seguiu as boas práticas identificadas na pesquisa bibliográfica e foi estruturado em sete etapas principais:

- Preparação do ambiente e setup.
- Extração automatizada de dados.
- Carga inicial em staging.
- Transformações com enriquecimento de dados.
- Modelagem dimensional (modelo estrela).
- Carga final em schema analítico.
- Orquestração com Apache Airflow.

Cada uma dessas fases é detalhada nos tópicos a seguir, com foco nas decisões técnicas, ferramentas utilizadas, estruturas criadas e aderência às boas práticas estabelecidas.

6.1 Diagnóstico e premissas do cenário simulado

Nesta etapa inicial da PoC, foi definido um cenário simulado representativo das operações de PMEs do setor de e-commerce. O objetivo foi criar um ambiente controlado

para a construção e teste de um pipeline de dados, utilizando uma base pública extraída do Kaggle. Tal escolha favoreceu tanto a reprodutibilidade quanto o caráter didático da aplicação.

O conjunto de dados selecionado contém cinco tabelas principais com registros transacionais de pedidos online: orders, order_items, customers, payments e products. A Tabela 9 resume a finalidade de cada tabela. Ao todo, a base contempla 89.316 pedidos, permitindo simular um fluxo completo de dados heterogêneos, característico de ambientes reais.

Tabela 7: Organização geral das tabelas utilizadas no estudo.

Tabela	Descrição
orders	Informações gerais dos pedidos, incluindo status e timestamps.
order_items	Itens de cada pedido, com detalhes de produto, preço e frete.
customers	Dados cadastrais e localização dos clientes.
payments	Detalhes dos pagamentos, tipo e valor transacionado.
products	Informações de produtos, categoria e dimensões.

O cenário simulado foi fundamental para validar o uso de ferramentas modernas como Python, PostgreSQL e Apache Airflow, permitindo o desenvolvimento de um pipeline ETL robusto e escalável. Além disso, a utilização de dados públicos reforça a proposta de replicabilidade, sendo condizente com a adoção de boas práticas para ambientes com recursos limitados, como é comum em PMEs.

6.2 Preparação do ambiente

A construção do pipeline foi conduzida em ambiente Linux (Ubuntu 22.04 via WSL 2), permitindo compatibilidade com ferramentas como Docker e Apache Airflow em um sistema Windows. Essa configuração garante estabilidade, isolamento e facilidade de replicação.

O ambiente foi estruturado com containers Docker, coordenados via docker-compose, integrando os seguintes serviços: PostgreSQL (armazenamento), Apache Airflow

(orquestração), Redis (fila de execução) e DBeaver (interface de banco). A estrutura do projeto incluiu variáveis de ambiente, diretórios persistentes e organização modular.

Essa abordagem se mostrou adequada para PMEs, pois favorece a manutenção, isolamento e implantação local com recursos limitados.

Os comandos e scripts utilizados nesta etapa encontram-se descritos no Apêndice A.

6.3 Criação do banco e uso do DBeaver

Após a preparação do ambiente, foi criado um banco de dados no PostgreSQL com o propósito de armazenar e organizar os dados extraídos, transformados e analisados durante a PoC. A criação inicial da base foi feita por meio do terminal WSL, utilizando comandos SQL básicos de inicialização e conexão.

A seguir, o gerenciamento do banco e das tabelas passou a ser realizado via DBeaver, uma interface gráfica de administração de bancos de dados que facilitou a inspeção dos dados e a execução de queries SQL. O DBeaver se mostrou uma ferramenta prática e acessível, contribuindo para a manipulação eficiente dos dados e acompanhamento da estrutura relacional ao longo das etapas do pipeline.

Durante esta etapa, foram criadas tabelas para armazenar os dados extraídos do dataset do Kaggle, bem como as tabelas transformadas resultantes das operações subsequentes. A interface visual do DBeaver permitiu verificar a integridade dos dados importados e validar a aplicação correta das etapas seguintes do fluxo.

Os comandos SQL de criação de base e configuração do DBeaver estão documentados no Apêndice B, de forma organizada e comentada.

6.4 Extração dos dados

A etapa de extração foi responsável por configurar o ambiente de projeto e realizar o download automatizado do conjunto de dados público selecionado. O processo evidenciou a flexibilidade do pipeline em integrar fontes externas por meio de APIs, com organização estruturada e rastreável dos artefatos gerados.

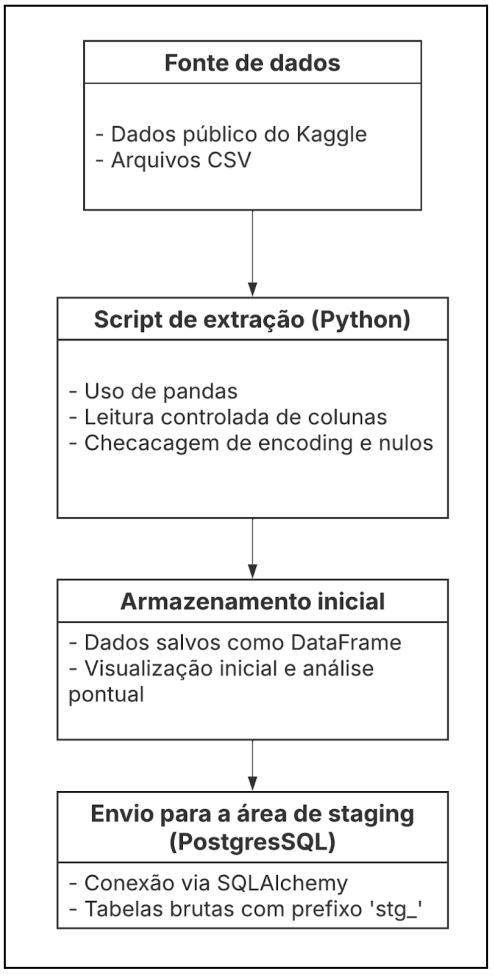
Inicialmente, foi definida uma estrutura de diretórios para centralizar os scripts, dados e logs do projeto. Em seguida, configurou-se um ambiente virtual Python com isolamento de dependências, permitindo maior controle sobre bibliotecas e reprodutibilidade. Foram instalados pacotes essenciais como pandas, sqlalchemy, psycopg2-binary e kaggle, voltados à manipulação de dados e à integração com a API do Kaggle.

A extração foi realizada por meio de script Python desenvolvido na pasta scripts, utilizando a biblioteca oficial kaggle. Após a configuração do token de autenticação de forma segura, o script acessou o dataset “ecommerce-order-dataset” e realizou o download automatizado dos arquivos .csv, armazenando-os na pasta local definida no projeto. Ao final da execução, uma mensagem de sucesso confirmou a operação.

Todos os comandos e scripts utilizados estão detalhados no Apêndice C.

A Figura 8 apresenta o fluxo de extração de dados realizado no início do pipeline, incluindo a fonte, leitura via Python e envio à área de staging.

Figura 8: Fluxo da etapa de extração de dados.



Fonte: Elaborado pelo autor.

6.5 Área de staging

Nesta etapa, os arquivos CSV brutos extraídos da plataforma Kaggle foram carregados em tabelas de staging no banco PostgreSQL. O principal objetivo dessa fase foi centralizar os dados de forma temporária, criando uma base confiável e organizada para as transformações futuras, enriquecimento e posterior modelagem dimensional.

Os arquivos carregados incluíram registros de pedidos, itens de pedidos, clientes, pagamentos e produtos. Esses arquivos estavam localizados em um diretório específico do projeto, conforme definido na estrutura de pastas da PoC. Para o carregamento, foi desenvolvido um script Python (staging_loader.py) utilizando as bibliotecas Pandas e SQLAlchemy, que realizou a leitura dos arquivos CSV e inserção nas tabelas do banco com nomenclatura padronizada (stg_).

Esse processo foi executado em ambiente virtual Python, com conexão estabelecida via string segura, e mapeamento automático dos arquivos para as tabelas correspondentes. A execução do script permitiu substituir tabelas anteriores (caso existentes), garantindo reprocessamento rápido quando necessário.

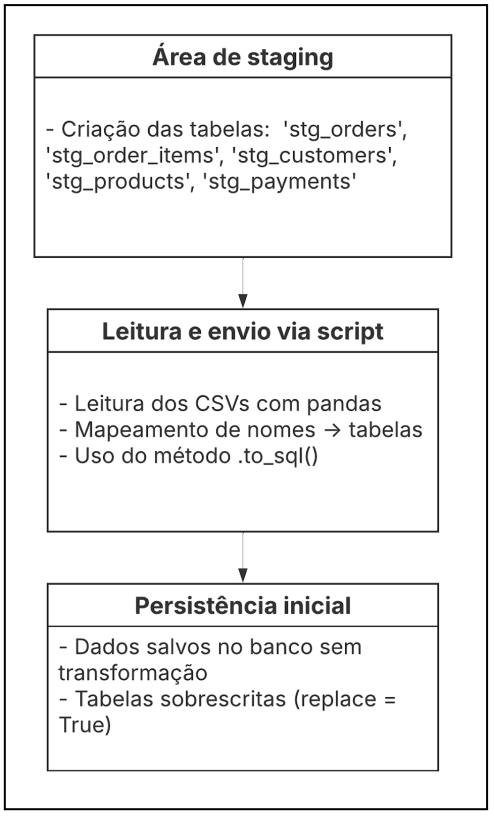
Apesar de a literatura técnica e a pesqu deste trabalho indicarem o uso de áreas de staging baseadas em armazenamento de objetos, como o Amazon S3, como uma boa prática em ambientes escaláveis (especialmente para sistemas em nuvem), optou-se por manter os dados temporários diretamente no PostgreSQL durante a Prova de Conceito. Essa decisão foi tomada por motivos de simplicidade operacional, facilidade de integração com o DBeaver e menor complexidade na orquestração local, respeitando o contexto de uma PME com infraestrutura limitada. A escolha não comprometeu os objetivos da PoC, mas manteve a clareza do fluxo e permitiu avaliação completa do ciclo ETL.

A validação foi feita através do DBeaver, onde as tabelas prefixadas com stg_ foram localizadas no schema public. Foram realizadas consultas básicas em SQL para verificação da integridade dos dados, como contagem de registros e inspeção visual de amostras.

Todos os scripts e definições utilizados para a carga na área de staging estão documentados no Apêndice D.

A Figura 9 apresenta o processo de envio dos dados extraídos para a área de staging no PostgreSQL, com a utilização de scripts Python e substituição controlada de tabelas.

Figura 9: Fluxo da carga dos dados na área de staging.



Fonte: Elaborado pelo autor.

6.6 Transformações aplicadas

Nesta etapa, os dados brutos armazenados nas tabelas de staging foram tratados e enriquecidos, preparando-os para posterior modelagem analítica (modelo estrela). As transformações seguiram boas práticas identificadas na literatura, com foco em modularidade, clareza e confiabilidade dos dados.

As principais operações realizadas incluíram:

- Leitura das tabelas de staging: Utilizando as bibliotecas pandas e SQLAlchemy, os dados foram importados diretamente do banco PostgreSQL, garantindo consistência entre as fontes.
- Limpeza e padronização: Foram removidas duplicidades nas tabelas orders e products, convertidas colunas de data para o tipo datetime, e padronizado o conteúdo da coluna order_status para letras minúsculas.
- Preenchimento de nulos: A coluna product_category_name da tabela de produtos teve os valores ausentes preenchidos com “Unknown”, assegurando coerência durante análises posteriores.
- Enriquecimento de dados: Foram calculadas métricas adicionais como:

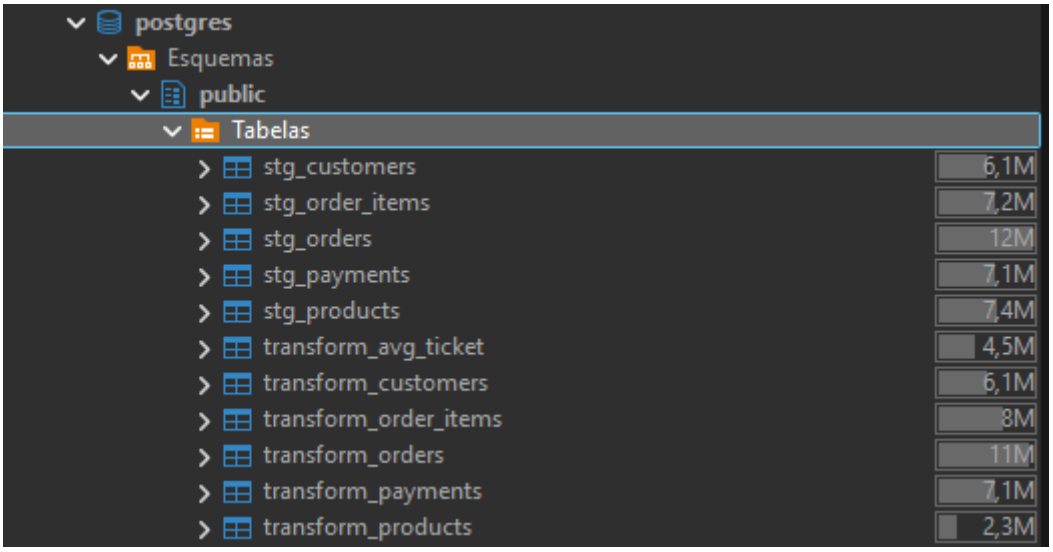
- delivery_days: diferença entre a data de entrega e a data da compra.
- is_delayed: indicador booleano de atraso, com base na comparação entre a data estimada e a data real de entrega.
- total_price: valor total de cada item (soma de preço e frete).
- items_count: quantidade de itens por pedido.

- Integração e agregação: Foi gerada uma tabela agregada com o total e a contagem de itens por pedido, posteriormente integrada à tabela orders. Além disso, foi calculado o ticket médio por cliente, a partir da média do valor total dos pedidos.
- Persistência no banco: As tabelas transformadas foram salvas no PostgreSQL com o prefixo transform_, substituindo versões anteriores, quando necessário. Essa prática facilita a rastreabilidade e o versionamento dos dados transformados.

Todos os scripts Python utilizados nesta etapa estão documentados no Apêndice E.

A validação final foi realizada no DBeaver, com verificação da existência e estrutura das tabelas transform_, conferência das colunas adicionais (delivery_days, is_delayed, order_total_value) e manutenção da granularidade na tabela transform_order_items.

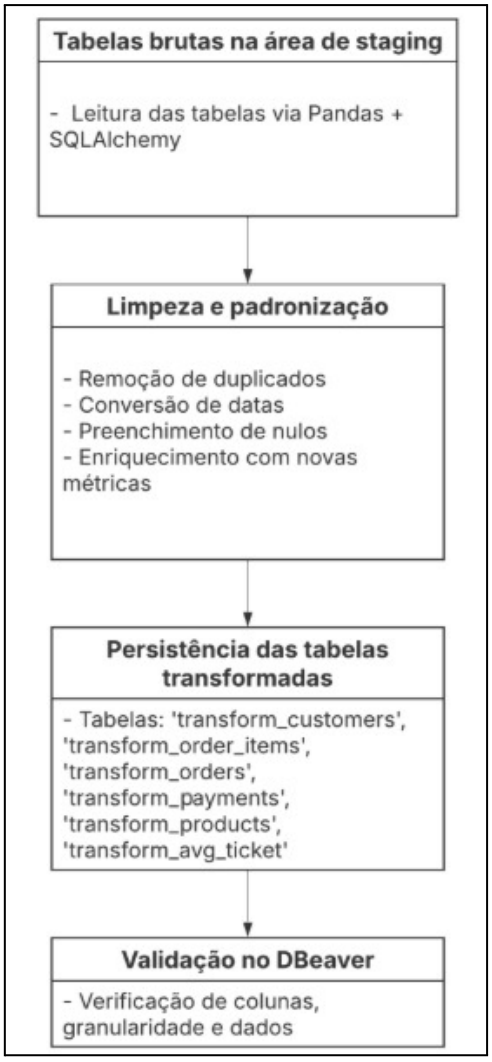
Figura 10: Tabelas com prefixo “transform_” localizadas no banco de dados.



Fonte: Interface do DBeaver (acesso em julho de 2025).

A Figura 11 apresenta o fluxo geral da transformação aplicada, incluindo a leitura das tabelas de staging, os tratamentos e enriquecimentos realizados, e a persistência final no banco de dados.

Figura 11: Fluxo das transformações aplicadas aos dados.



Fonte: Elaborado pelo autor.

Essa etapa consolidou os dados em um formato limpo e confiável, garantindo sua qualidade antes da carga final e posterior análise. O uso de operações modulares e o controle por scripts versionados favorecem a escalabilidade do pipeline, sendo compatíveis com as necessidades de pequenas e médias empresas que buscam soluções eficientes e de fácil manutenção.

6.7 Modelagem dimensional (modelo estrela)

A modelagem dimensional teve como objetivo estruturar os dados transformados em um formato analítico, facilitando a geração de relatórios e análises em ferramentas de BI. Optou-se pelo modelo estrela, por sua simplicidade e aderência a contextos de PMEs, com uma tabela fato central e três dimensões.

A tabela fato criada (fact_orders) consolidou os principais indicadores de negócio a partir das tabelas transformadas, incluindo valores de pedidos, número de itens, atrasos e prazos de entrega. A granularidade foi definida no nível do pedido, considerando as limitações do dataset e a busca por um modelo enxuto.

As dimensões geradas foram:

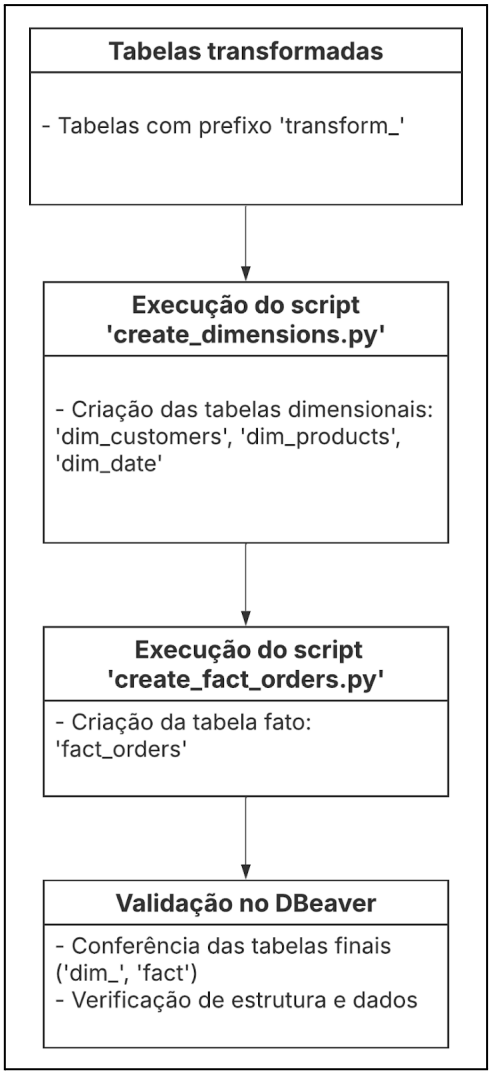
- dim_customers: com informações geográficas (cidade e estado).
- dim_products: com atributos de categoria, peso e dimensões.
- dim_date: com dados derivados de tempo (dia, mês, ano, dia da semana).

A criação das tabelas foi realizada por scripts em Python com o uso de pandas e SQLAlchemy, armazenando os dados no PostgreSQL. A validação foi feita via DBeaver, garantindo estrutura adequada e integridade dos dados.

Os scripts de criação das tabelas dimensionais e da tabela fato estão detalhados no Apêndice F.

A Figura 12 apresenta o fluxo de criação das tabelas dimensionais e da tabela fato a partir das tabelas transformadas.

Figura 12: Etapas da modelagem dimensional.



Fonte: Elaborado pelo autor.

6.8 Carga final e análise

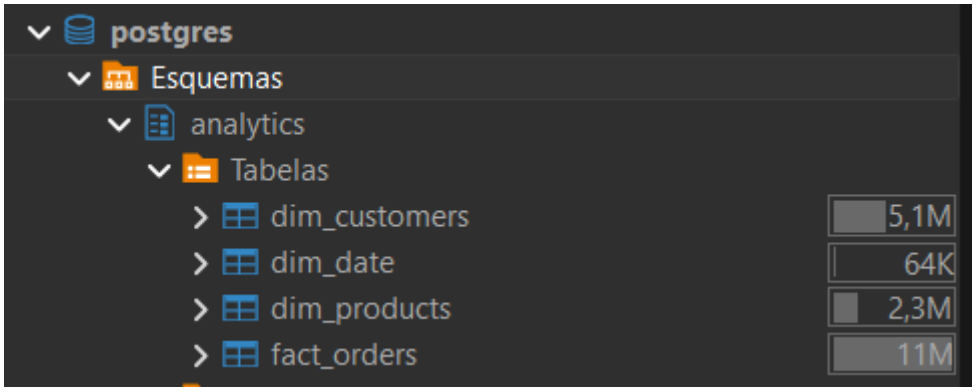
A etapa de carga final teve como objetivo consolidar as tabelas analíticas (fato e dimensões) em um schema exclusivo no PostgreSQL, denominado analytics. Essa prática contribui para a organização lógica do repositório, facilita o gerenciamento de permissões e isola os dados prontos para análise, respeitando boas práticas de governança e arquitetura de dados.

A carga foi realizada em três etapas principais:

- Criação do schema analytics.
- Movimentação das tabelas finais (dim_customers, dim_products, dim_date e fact_orders) do schema padrão para o novo schema analítico, via comandos ALTER TABLE SET SCHEMA.

- Validação no DBeaver, verificando a existência das tabelas, estrutura esperada, integridade dos registros e presença de chaves conceituais.

Figura 13: Tabelas carregas no schema analytics.



Fonte: Interface do DBeaver (acesso em julho de 2025).

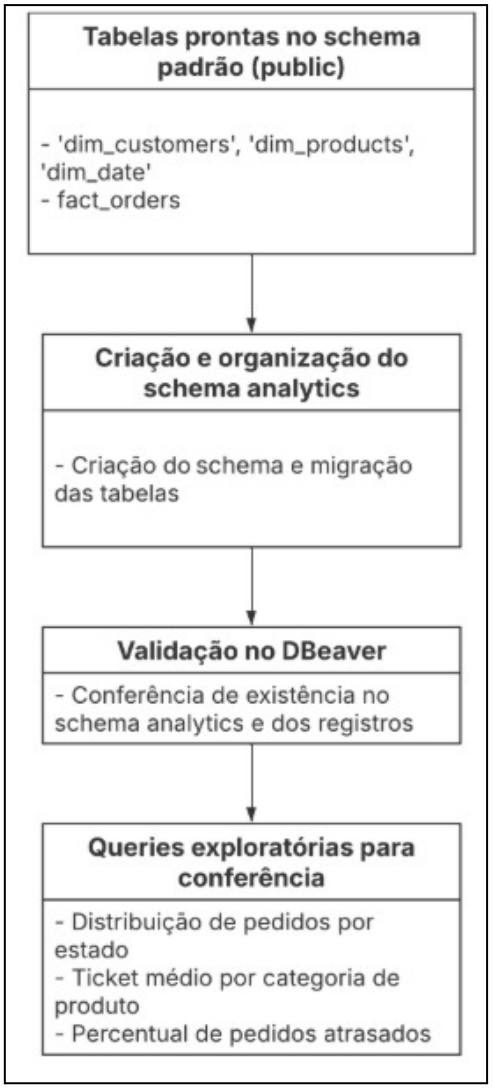
Por fim, foram executadas queries exploratórias para validar a integridade das relações e a consistência dos dados. Entre os exemplos, destacam-se análises de distribuição de pedidos por estado, cálculo de ticket médio por categoria e percentual de atrasos nas entregas.

Essas consultas demonstraram a aderência entre as tabelas, confirmando a eficácia do pipeline na preparação de dados analíticos reutilizáveis, tanto para SQL direto quanto para uso em ferramentas de BI.

Os comandos utilizados nesta etapa estão detalhados do Apêndice G.

A Figura 14 apresenta o processo de carga das tabelas analíticas em um schema dedicado (analytics) no banco PostgreSQL e as queries exploratórias realizadas para validação.

Figura 14: Carga final e análise no schema analytics.



Fonte: Elaborado pelo autor.

6.9 Orquestração e automação com Airflow

A etapa de orquestração teve como objetivo automatizar o pipeline ETL utilizando o Apache Airflow, garantindo a execução encadeada das tarefas de extração, staging, transformação e carga no banco PostgreSQL.

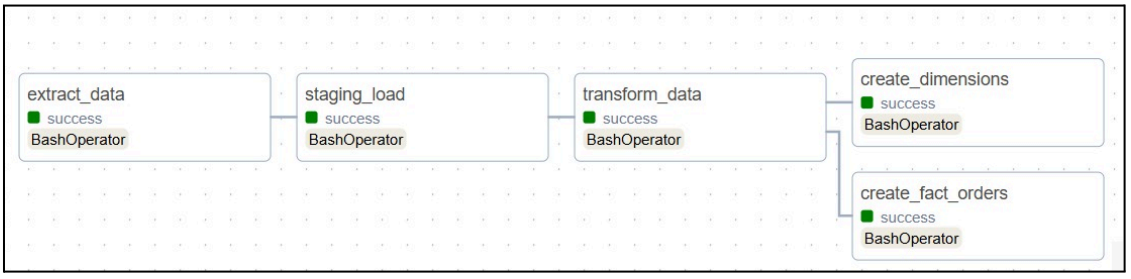
Para isso, foi utilizado um ambiente Docker com containers específicos para o Airflow e para o PostgreSQL. A DAG desenvolvida, nomeada etl_project_pipeline, foi configurada com execução manual (schedule_interval=None) e composta pelas seguintes tarefas:

- extract_data: responsável pela extração do dataset via API do Kaggle, com instalação prévia do pacote kaggle no container.
- staging_load: realiza a carga dos dados brutos para a área de staging.

- transform_data: aplica as transformações necessárias, gerando tabelas intermediárias com dados tratados.
- create_dimensions e create_fact_orders: criam, respectivamente, as tabelas dimensionais e de fatos, com base nas transformações realizadas. Essas duas tarefas são executadas em paralelo.

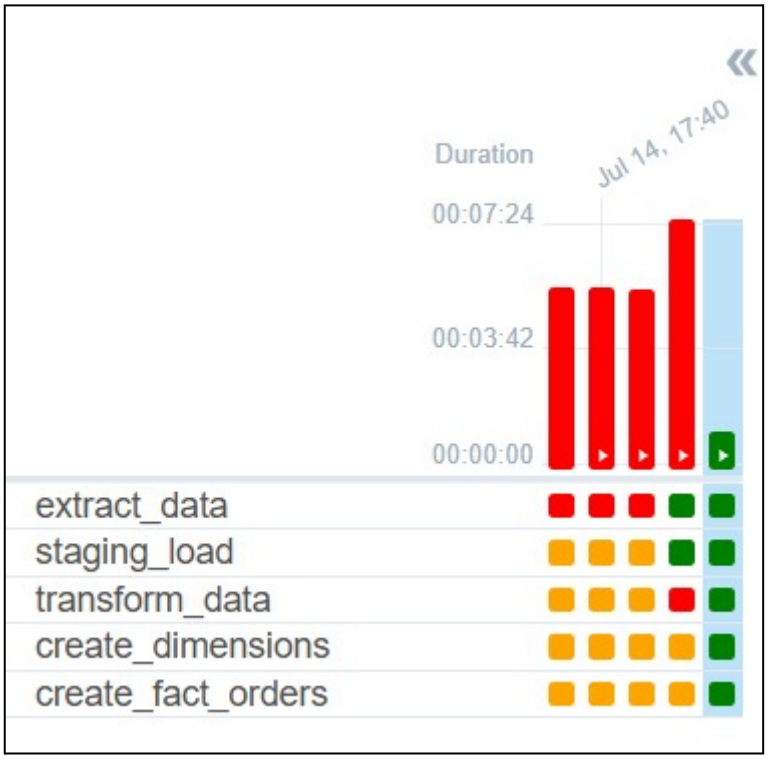
A execução da DAG foi realizada por meio da interface do Airflow (acessível via navegador), permitindo o acompanhamento gráfico do fluxo e das durações de cada tarefa, como ilustrado na Figura 15 e Figura 16. O controle dos containers foi feito via docker-compose, garantindo a persistência do banco e o isolamento do ambiente de orquestração.

Figura 15: Visualização em grafo da DAG no Airflow.



Fonte: Interface do Airflow (acesso em julho de 2025).

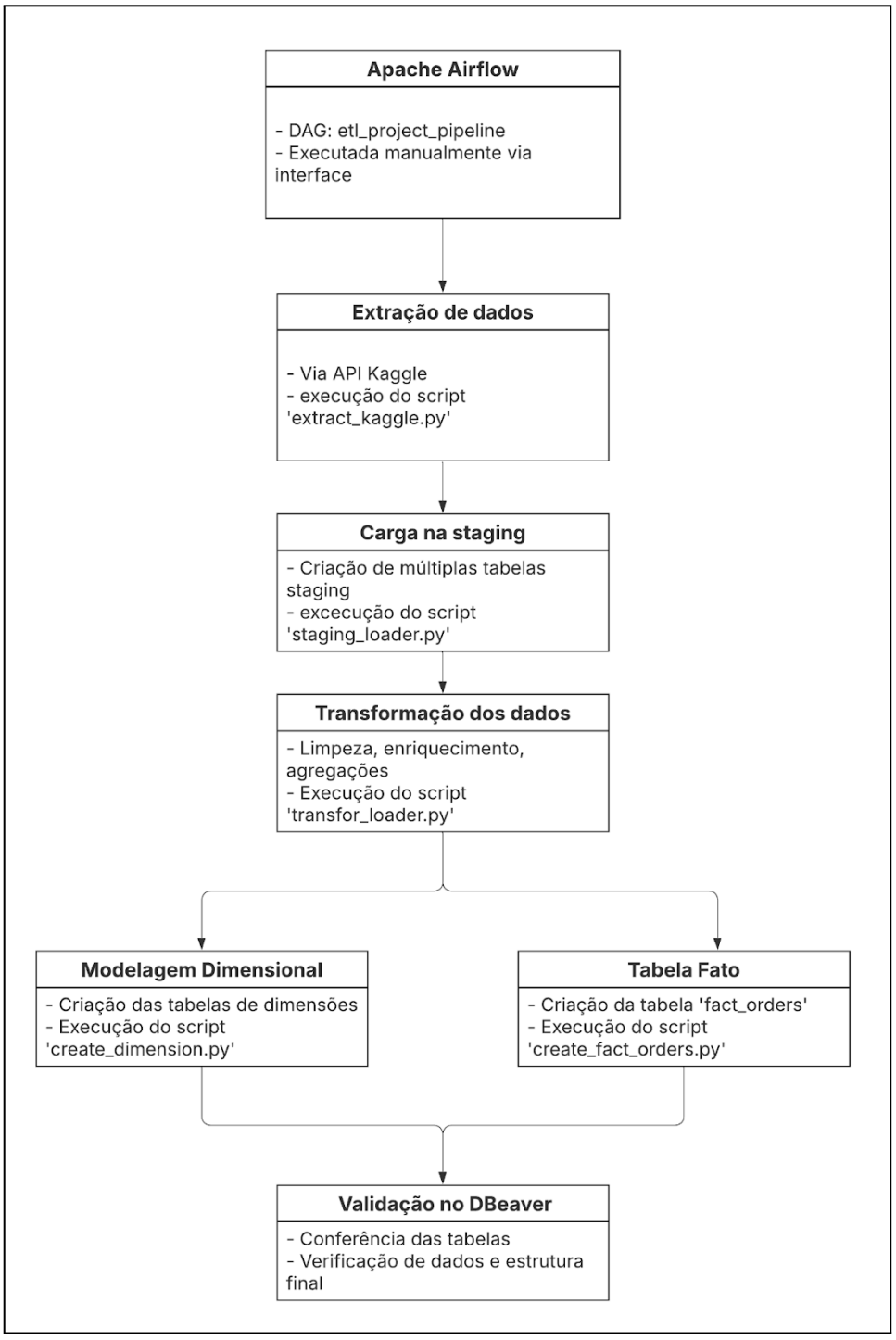
Figura 16: Visualização do gráfico de duração e status das tarefas no Airflow.



Fonte: Interface do Airflow (acesso em julho de 2025).

Ao final da execução, as tabelas geradas puderam ser validadas diretamente no DBeaver, o que confirmou a integridade e o encadeamento adequado de todas as etapas do pipeline. A definição completa da DAG, contendo a estrutura de tarefas, dependências e parâmetros configurados, está apresentada no Apêndice H. A Figura 17 apresenta o fluxo adotado nesta etapa para a PoC.

Figura 17: Orquestração das etapas do pipeline no Apache Airflow.



Fonte: Elaborado pelo autor.

6.10 Validação, visualização e governança

Após a carga final dos dados no schema analítico, foram realizadas validações e análises exploratórias com o objetivo de garantir a integridade das informações e a conformidade com os objetivos definidos para o pipeline.

A validação foi conduzida diretamente no DBeaver, por meio de consultas SQL nos dados consolidados. Essa etapa incluiu:

- Conferência do volume de registros por tabela.
- Análise de consistência dos joins entre tabelas fato e dimensões.
- Verificação de nulos, duplicidades e anomalias em campos-chave.

Na sequência, as tabelas foram exploradas também sob uma perspectiva analítica, com a execução de queries que simulam cenários reais de negócios. Entre os exemplos, destacam-se:

- Análise da distribuição de pedidos por estado.
- Cálculo do ticket médio por categoria de produto.
- Percentual de entregas com atraso.

Essas consultas reforçaram a qualidade das transformações aplicadas e evidenciaram a capacidade do pipeline em sustentar análises mais robustas, tanto diretamente via SQL quanto por ferramentas externas de BI.

Além da validação técnica, a separação das tabelas em um schema exclusivo (analytics) também viabilizou uma governança mais eficaz. Isso permitiu o controle de acessos, organização lógica das informações e isolamento do ambiente analítico em relação às operações do dia a dia.

6.11 Aderência às boas práticas e análise crítica da implementação

A construção do pipeline ETL nesta PoC demonstrou a viabilidade de aplicar boas práticas de engenharia de dados mesmo em contextos com recursos limitados, como é comum em PMEs. A estrutura modular adotada com etapas claras de extração, staging, transformação, modelagem dimensional, carga final e orquestração, permitiu ganhos em clareza, manutenção e escalabilidade.

Dentre os pontos positivos observados, destacam-se:

- Simplicidade na replicação do ambiente, viabilizada pelo uso do Docker e do Apache Airflow.

- Clareza no fluxo de dados, com separação entre dados brutos e tratados (staging → analytics).
- Rastreabilidade e controle, proporcionados pela orquestração com Airflow, que mitiga erros manuais e facilita o monitoramento de execuções.
- Organização e reutilização do código, com scripts centralizados e versionáveis.
- Adoção de modelo dimensional em estrela, que facilita análises exploratórias e consultas por ferramentas de BI.

Apesar disso, a implementação também revelou limitações. A principal delas diz respeito à base de dados utilizada (Kaggle), que não reflete todas as nuances e desafios operacionais de uma PME real, como consistência, latência ou qualidade dos dados. Além disso, a PoC não contemplou mecanismos robustos de monitoramento contínuo (observabilidade) nem testes automatizados.

No tocante à aderência às boas práticas, a solução se alinha aos princípios de modularidade, automação, separação lógica por schema, controle de dependências e organização dos scripts. A configuração da DAG de orquestração ilustra esse alinhamento. De maneira geral, a PoC atingiu os objetivos propostos, demonstrando que é possível construir pipelines de dados robustos, versionáveis e confiáveis mesmo em ambientes com infraestrutura limitada. Para cenários reais, recomenda-se avançar com mecanismos de validação contínua, monitoramento ativo e integração com ferramentas de visualização em tempo real.

7. CONCLUSÃO E RECOMENDAÇÕES

Este trabalho teve como objetivo principal mapear, sistematizar e aplicar boas práticas para a construção de pipelines de dados em PMEs, com foco específico nos processos de ETL. Partindo da constatação de que a adoção dessas tecnologias ainda encontra barreiras significativas nas PMEs, como as limitações técnicas, escassez de recursos humanos qualificados e ausência de diretrizes claras, a pesquisa buscou oferecer uma contribuição prática por meio da combinação entre pesquisa bibliográfica e implementação de uma Prova de Conceito.

A pesquisa bibliográfica revelou um conjunto de desafios recorrentes enfrentados por PMEs na integração de dados, destacando-se a heterogeneidade das fontes, a falta de automação, os custos elevados de ferramentas tradicionais e a fragilidade das políticas de

governança de dados. Por outro lado, abordagens baseadas em modularidade, orquestração automatizada, uso de ferramentas open-source e modelagem dimensional simplificada são eficazes e viáveis mesmo em contextos com infraestrutura restrita.

A PoC desenvolvida validou, em ambiente simulado, a aplicação dessas boas práticas por meio de ferramentas como Python, PostgreSQL, Docker e Apache Airflow. O pipeline construído foi capaz de realizar a extração, transformação e carga de dados em um fluxo automatizado, modular e rastreável. A estruturação da arquitetura em camadas (staging e analytics) e a adoção do modelo estrela para suporte analítico demonstraram que é possível construir soluções escaláveis e reprodutíveis, compatíveis com as demandas de PMEs.

A contribuição prática deste trabalho reside na demonstração de que, com planejamento, adoção de boas práticas e uso inteligente de tecnologias acessíveis, é viável implementar pipelines de dados robustos e sustentáveis. Essa constatação pode servir de guia técnico e estratégico para outras PMEs interessadas em fortalecer sua cultura data-driven e melhorar sua capacidade analítica.

Entretanto, algumas limitações devem ser reconhecidas. A PoC foi desenvolvida com base em um dataset simulado e, embora representativo, não contempla todas as variáveis e desafios de ambientes reais, como dados inconsistentes, volumes imprevisíveis e variações de latência. Além disso, não foram abordados mecanismos mais avançados de monitoramento, validação automatizada ou integração contínua (CI/CD) do pipeline, que são relevantes em ambientes de produção.

Trabalhos futuros incluem:

- Replicação da PoC em ambientes reais de PMEs, com dados operacionais autênticos;
- Expansão da arquitetura para contemplar mecanismos de observabilidade e testes automatizados;
- Análise comparativa entre diferentes ferramentas de orquestração
- Investigação da integração entre pipelines de dados e plataformas de Business Intelligence (BI), com foco na construção de dashboards e indicadores adaptados à realidade das PMEs.

Por fim, recomenda-se que futuras pesquisas explorem a interface entre pipelines de dados e decisões de negócio, investigando como a maturidade analítica pode ser integrada aos processos estratégicos das PMEs. A consolidação de uma cultura orientada a dados depende não apenas de infraestrutura técnica, mas de uma mudança organizacional mais ampla, que valorize a informação como ativo central para a inovação e a competitividade.

REFERÊNCIAS

JOHNSON, Ebunoluwa et al. Developing scalable data solutions for small and medium enterprises: challenges and best practices. International Journal of Management & Entrepreneurship Research, v. 6, n. 6, p. 1910-1935, jun. 2024.

CHIKHALKAR, Akshay et al. A data pipeline concept for digitizing services in small and medium-sized companies. International Journal on Informatics Visualization (JOIV), v. 9, n. 1, p. 333-341, jan. 2025.

ALSIBHAWI, Ibrahim Abdusalam Abubaker; YAHAYA, Jamaiah Binti; MOHAMED, Hazura Binti. Business intelligence adoption for small and medium enterprises: conceptual framework. Applied Sciences, v. 13, n. 7, p. 4121, 2023.

TAWIL, Abdel-Rahman H. et al. Trends and challenges towards an effective data-driven decision making in UK SMEs: case studies and lessons learnt from the analysis of 85 SMEs. Big Data and Cognitive Computing, v. 8, n. 7, art. 79, jul. 2024.

DAPKUTE, Austeja et al. Digital twin data management: framework and performance metrics of cloud-based ETL system. Machines, v. 12, n. 2, art. 130, 2024.

DINESH, Lina; DEVI, K. Gayathri. An efficient hybrid optimization of ETL process in data warehouse of cloud architecture. Journal of Cloud Computing: Advances, Systems and Applications, v. 13, art. 12, 2024.

RAGHAV; BARANI; VIJAY RAM. ETL: from design to deployment. Golden Sun-Rise International Journal of Multidisciplinary on Science and Management, v. 1, n. 3, p. 1-13, jul.-set. 2024.

KHAN, Bilal et al. An overview of ETL techniques, tools, processes and evaluations in data warehousing. Journal on Big Data, v. 6, 26 jan. 2024.

EETI, Shanmukha. Efficient ETL processes: a comparative study of Apache Airflow vs. traditional methods. Journal of Emerging Technologies and Innovative Research (JETIR), v. 9, n. 8, p. 100-112, ago. 2022.

ZARATE, Gorka et al. Evolution of extract-transform-load (ETL) processes towards data product pipelines. In: Proceedings of the 4th Eclipse Security, AI, Architecture and Modelling Conference on Data (eSAAM 2024), Mainz, Germany, out. 2024. ACM, New York, p. 1-8.

MEHMOOD, Erum; ANEES, Tayyaba. An efficient hybrid optimization of ETL process in data warehouse of cloud architecture. IEEE Access, v. 8, p. 119325-119340, 2020.

NWOKEJI, Joshua C.; MATOVU, Richard. A systematic literature review on big data extraction, transformation and loading (ETL). In: Intelligent Computing. Proceedings of the 2021 Computing Conference, v. 2, p. 308-324. Springer, 2021.

ROZONY, Farhana Zaman et al. A systematic review of big data integration challenges and solutions for heterogeneous data sources. Academic Journal on Business, Administration, Innovation & Sustainability, v. 4, n. 4, p. 1-18, out. 2024.

ALI, Syed Muhammad Fawad; WREMBEL, Robert. From conceptual design to performance optimization of ETL workflows: current state of research and open problems. The VLDB Journal, v. 26, p. 777-801, 2017.

BECKER, Michel et al. Big data em micro e pequenas empresas: uma revisão sistemática. Revista GeSec, v. 14, n. 3, p. 3420-3442, 2023.

YASMIN, Jerin et al. An empirical study of developers' challenges in implementing Workflows as Code: a case study on Apache Airflow. Journal of Systems and Software, v. 219, art. 112248, jan. 2025.

MACHIREDDY, Jeshwanth Reddy. Data quality management and performance optimization for enterprise-scale ETL pipelines in modern analytical ecosystems. Journal of Data Science, Predictive Analytics, and Big Data Applications, v. 8, n. 7, p. 1-26, 2023.

BERISHA, Gentrit; SHIROKA PULA, Justina. Defining small and medium enterprises: a critical review. Academic Journal of Business, Administration, Law and Social Sciences, v. 1, n. 1, p. 17-28, mar. 2015.

WIJAYA, Willsen et al. Implementation of data warehouse and star schema for optimizing property business decision making. G-Tech: Jurnal Teknologi Terapan, v. 8, n. 2, p. 1242-1250, abr. 2024.

MANDALA, Nishanth Reddy. Advances in distributed storage systems for big data. Journal of Mathematical & Computer Applications, v. 2, n. 2, p. 1-8, 2023.

BANCO NACIONAL DE DESENVOLVIMENTO ECONÔMICO E SOCIAL (BNDES). Porte de empresa. 2022. Disponível em: https://www.bndes.gov.br/wps/portal/site/home/financiamento/guia/porte-de-empresa .

GIL, Antônio Carlos. Métodos e técnicas de pesquisa social. 7. ed. São Paulo: Atlas, 2019.

APÊNDICES

A Apêndice 1: Preparação do ambiente de execução

A.1 Instalação do Ubuntu via WSL e preparação inicial

```
wsl --install -d Ubuntu

wsl

sudo apt update
sudo apt upgrade -y
```

A.2 Instalação e configuração do Docker

```
sudo apt install -y docker.io docker-compose
sudo service docker start
sudo usermod -aG docker $USER
newgrp docker
```

A.3 Estrutura do projeto e arquivos iniciais

```
mkdir ~/airflow_docker
cd ~/airflow_docker

curl -LfO
'https://airflow.apache.org/docs/apache-airflow/2.7.0/docker-c
ompose.yaml'

mkdir -p ./dags ./logs ./plugins ./config

echo -e "AIRFLOW_UID=$(id -u)" > .en
```

A.4 Inicialização e execução dos serviços e acesso à interface do Airflow

```
docker-compose up airflow-init
```

```
docker-compose up
http://localhost:8080
```

B Apêndice 2: Criação do banco e configuração com DBeaver

B.1 Instalação e configuração do PostgreSQL

```
sudo apt update
sudo apt install -y postgresql postgresql-contrib
sudo service postgresql start

sudo -u postgres psql

CREATE DATABASE postgres;
CREATE USER guilherme WITH PASSWORD 'guilherme123';
GRANT ALL PRIVILEGES ON DATABASE postgres TO guilherme;
\q
```

B.2 Conexão ao PostgreSQL no DBeaver

Parâmetros utilizados:

- Host: localhost
- Porta: 5432
- Database: postgres
- Usuário: guilherme
- Senha: guilherme123

C Apêndice 3: Extração dos dados

C.1 Estrutura de diretórios do projeto e criação de ambiente virtual

```
mkdir -p ~/etl_project/scripts ~/etl_project/data
~/etl_project/logs
cd ~/etl_project

sudo apt install python3.12-venv
python3.12 -m venv venv
source venv/bin/activate

pip install pandas sqlalchemy psycopg2-binary kaggle
```

C.2 Criação do script de extração com API Kaggle

```
cd ~/etl_project/scripts
nano extract_kaggle.py
```

```
from kaggle.api.kaggle_api_extended import KaggleApi
api = KaggleApi()
api.authenticate()
dataset = 'bytdatit/ecommerce-order-dataset'
output_path = '../data'
api.dataset_download_files(dataset, path=output_path,
unzip=True)
print("Download concluído. Dados salvos em", output_path)
```

C.3 Configuração do token de autenticação do Kaggle execução do script de extração

```
mkdir -p ~/.kaggle
cp /mnt/c/Users/guilh/Downloads/kaggle.json ~/.kaggle/
chmod 600 ~/.kaggle/kaggle.json
```

```
dataset = 'bytdatit/ecommerce-order-dataset'
output_path = '../data

cd ~/etl_project/scripts
python extract_kaggle.py
```

D Apêndice 4: Área de Staging (Pré-transformação)

D.1 Organização e criação do script de carga

```
mkdir ~/etl_project/scripts
cd ~/etl_project/scripts
nano staging_loader.py
```

```
import pandas as pd
from sqlalchemy import create_engine

# Configuração do banco
user = 'guilherme'
password = 'guilherme123'
host = 'localhost'
port = '5432'
database = 'postgres'

engine =
create_engine(f'postgresql+psycopg2://{user}:{password}@{host}
:{port}/{database}')

# Mapeamento arquivos - tabelas
files_tables = {
    'df_Orders.csv': 'stg_orders',
    'df_OrderItems.csv': 'stg_order_items',
    'df_Customers.csv': 'stg_customers',
    'df_Payments.csv': 'stg_payments',
    'df_Products.csv': 'stg_products'
}

for file_name, table_name in files_tables.items():
    print(f"Carregando {file_name} -> {table_name}")
    df = pd.read_csv(f'../data/Ecommerce Order
Dataset/train/{file_name}')
```

```
    df.to_sql(table_name, engine, if_exists='replace',
index=False)
    print(f"Tabela {table_name} criada com sucesso")

print("Todas as tabelas de staging foram carregadas no banco
PostgreSQL")
```

D.2 Execução do script e validação das tabelas no DBeaver

```
python staging_loader.py
```

```
SELECT COUNT(*) FROM stg_orders;
SELECT * FROM stg_customers LIMIT 10;
```

E Apêndice 5: Transformações aplicadas

E.2 Criação do script 'transform_loader.py’

```
import pandas as pd
from sqlalchemy import create_engine

# Configuração do banco de dados
user = 'guilherme'
password = 'guilherme123'
host = 'localhost'
port = '5432'
database = 'postgres'

engine =
create_engine(f'postgresql+psycopg2://{user}:{password}@{host}
:{port}/{database}')

# Leitura das tabelas de staging
orders = pd.read_sql('SELECT * FROM stg_orders', engine)
order_items = pd.read_sql('SELECT * FROM stg_order_items',
engine)
customers = pd.read_sql('SELECT * FROM stg_customers', engine)
payments = pd.read_sql('SELECT * FROM stg_payments', engine)
products = pd.read_sql('SELECT * FROM stg_products', engine)

# Limpeza e normalização
orders = orders.drop_duplicates()
products = products.drop_duplicates()
orders['order_status'] = orders['order_status'].str.lower()

orders['order_purchase_timestamp'] =
pd.to_datetime(orders['order_purchase_timestamp'])
orders['order_approved_at'] =
pd.to_datetime(orders['order_approved_at'])
```

```
orders['order_delivered_timestamp'] =
pd.to_datetime(orders['order_delivered_timestamp'])
orders['order_estimated_delivery_date'] =
pd.to_datetime(orders['order_estimated_delivery_date'])

products['product_category_name'] =
products['product_category_name'].fillna("Unknown")

# Enriquecimento
orders['delivery_days'] = (orders['order_delivered_timestamp']
- orders['order_purchase_timestamp']).dt.days
orders['is_delayed'] = orders['order_delivered_timestamp'] >
orders['order_estimated_delivery_date']

order_items['total_price'] = order_items['price'] +
order_items['shipping_charges']

agg_orders = order_items.groupby('order_id').agg({
    'total_price': 'sum',
    'product_id': 'count'
}).rename(columns={
    'total_price': 'order_total_value',
    'product_id': 'items_count'
}).reset_index()

orders = pd.merge(orders, agg_orders, on='order_id',
how='left')

avg_ticket =
orders.groupby('customer_id')['order_total_value'].mean().rese
t_index()
avg_ticket = avg_ticket.rename(columns={'order_total_value':
'avg_ticket_value'})
```

```
# Salvar tabelas transformadas
orders.to_sql('transform_orders', engine, if_exists='replace',
index=False)
products.to_sql('transform_products', engine,
if_exists='replace', index=False)
customers.to_sql('transform_customers', engine,
if_exists='replace', index=False)
payments.to_sql('transform_payments', engine,
if_exists='replace', index=False)
avg_ticket.to_sql('transform_avg_ticket', engine,
if_exists='replace', index=False)
order_items.to_sql('transform_order_items', engine,
if_exists='replace', index=False)

print("Transformação concluída e tabelas salvas no banco
PostgreSQL")
```

E.2 Execução do script

```
cd ~/etl_project/scripts
python transform_loader.py
```

F Apêndice 6: Modelagem dimensional

F.1 Criação das dimensões

```
import pandas as pd
from sqlalchemy import create_engine

# Conexão com o banco
user = 'guilherme'
password = 'guilherme123'
host = 'localhost'
port = '5432'
database = 'postgres'

engine =
create_engine(f'postgresql+psycopg2://{user}:{password}@{host}
:{port}/{database}')

# Dimensão Clientes
customers = pd.read_sql('SELECT * FROM transform_customers',
engine)
dim_customers = customers[['customer_id', 'customer_city',
'customer_state']].drop_duplicates()
dim_customers.to_sql('dim_customers', engine,
if_exists='replace', index=False)
print("Dimensão Clientes criada")

# Dimensão Produtos
products = pd.read_sql('SELECT * FROM transform_products',
engine)
dim_products = products[['product_id',
'product_category_name',
                         'product_weight_g',
'product_length_cm',
```

```
                        'product_height_cm',
'product_width_cm']].drop_duplicates()
dim_products.to_sql('dim_products', engine,
if_exists='replace', index=False)
print("Dimensão Produtos criada")

# Dimensão Datas
orders = pd.read_sql('SELECT * FROM transform_orders', engine)
dates =
pd.to_datetime(orders['order_purchase_timestamp']).dt.date.uni
que()
df_dates = pd.DataFrame({'date': pd.to_datetime(dates)})
df_dates['date'] =
df_dates['date'].dt.strftime('%Y%m%d').astype(int)
df_dates['day'] =
df_dates['date'].astype(str).str[-2:].astype(int)
df_dates['month'] =
df_dates['date'].astype(str).str[4:6].astype(int)
df_dates['year'] =
df_dates['date'].astype(str).str[:4].astype(int)
df_dates['day_of_week'] = pd.to_datetime(df_dates['date'],
format='%Y%m%d').dt.dayofweek + 1
df_dates.to_sql('dim_date', engine, if_exists='replace',
index=False)
print("Dimensão Data criada com sucesso")
```

F.2 Criação da tabela fato

```
import pandas as pd
from sqlalchemy import create_engine

# Conexão
user = 'guilherme'
password = 'guilherme123'
```

```
host = 'localhost'
port = '5432'
database = 'postgres'

engine =
create_engine(f'postgresql+psycopg2://{user}:{password}@{host}
:{port}/{database}')

# Carregar transform_orders
orders = pd.read_sql('SELECT * FROM transform_orders', engine)
orders_fact = orders[['order_id', 'customer_id',
                      'order_purchase_timestamp',
'order_delivered_timestamp',
                      'order_estimated_delivery_date',
'order_total_value',
                      'items_count', 'delivery_days',
'is_delayed']].copy()

# Adicionar payment_value
payments = pd.read_sql('SELECT order_id, payment_value FROM
transform_payments', engine)

# Adicionar shipping_value
order_items = pd.read_sql('SELECT order_id, shipping_charges
FROM transform_order_items', engine)
shipping_agg =
order_items.groupby('order_id')['shipping_charges'].sum().rese
t_index()

# Merge final
orders_fact = orders_fact.merge(payments, on='order_id',
how='left')
orders_fact = orders_fact.merge(shipping_agg, on='order_id',
how='left')
```

```
# Salvar tabela fato
orders_fact.to_sql('fact_orders', engine, if_exists='replace',
index=False)
print("Tabela fato criada com sucesso")
```

F.3 Execução dos scripts no terminal

```
cd ~/etl_project/scripts
nano create_dimensions.py     # Colar conteúdo das dimensões
nano create_fact_orders.py    # Colar conteúdo da fato

source ~/etl_project/venv/bin/activate
python create_dimensions.py
python create_fact_orders.py
```

G Apêndice 7: Carga final

G.1 Criação do schema analytics e movimentação das tabelas finais

```
CREATE SCHEMA IF NOT EXISTS analytics;
```

```
ALTER TABLE dim_customers SET SCHEMA analytics;
ALTER TABLE dim_products SET SCHEMA analytics;
ALTER TABLE dim_date SET SCHEMA analytics;
ALTER TABLE fact_orders SET SCHEMA analytics;
```

H Apêndice 7: Orquestração e Automação com Airflow

H.1 Preparação e configuração do ambiente Docker + Airflow

```
user = 'guilherme'
password = 'guilherme123'
host = 'postgres'  # nome do serviço no docker-compose
port = '5432'
database = 'airflow'

engine =
create_engine(f'postgresql+psycopg2://{user}:{password}@{host}
:{port}/{database}')
```

```
docker exec -it airflow_docker_airflow-webserver_1 bash
pip install kaggle
exit

docker exec -it airflow_docker_airflow-worker_1 bash
pip install kaggle
exit
```

H.2 DAG

```
from airflow import DAG
from airflow.operators.bash import BashOperator
from datetime import datetime

default_args = {
    'owner': 'guilherme',
    'start_date': datetime(2025, 7, 11),
    'retries': 1
}
```

```
dag = DAG(
    'etl_project_pipeline',
    default_args=default_args,
    description='Pipeline ETL para Ecommerce',
    schedule_interval=None,
    catchup=False
)

extract = BashOperator(
    task_id='extract_data',
    bash_command='python
/opt/airflow/dags/scripts/extract_kaggle.py',
    dag=dag,
)

staging = BashOperator(
    task_id='staging_load',
    bash_command='python
/opt/airflow/dags/scripts/staging_loader.py',
    dag=dag,
)

transform = BashOperator(
    task_id='transform_load',
    bash_command='python
/opt/airflow/dags/scripts/transform_loader.py',
    dag=dag,
)

dimensions = BashOperator(
    task_id='create_dimensions',
    bash_command='python
/opt/airflow/dags/scripts/create_dimensions.py',
```

```
    dag=dag,
)

fact_orders = BashOperator(
    task_id='create_fact_orders',
    bash_command='python
/opt/airflow/dags/scripts/create_fact_orders.py',
    dag=dag,
)

extract >> staging >> transform >> [dimensions, fact_orders]
```