# Projeto de TCC - Pipeline ETL Auto-Corretivo com IA

## Título

**Pipeline ETL Auto-Corretivo com Inteligência Artificial para Qualidade de Dados**

## 1. Introdução

Com o crescimento exponencial da quantidade de dados gerados diariamente, garantir a **qualidade dos dados** se tornou um dos maiores desafios no cenário de tecnologia. Dados inconsistentes, incompletos ou com formatos diferentes inviabilizam análises, relatórios e tomadas de decisão.

Este projeto apresenta um **Pipeline ETL Auto-Corretivo com IA**, capaz de:

* Detectar automaticamente inconsistências e anomalias nos dados
* Corrigir padrões divergentes com base na maioria
* Reportar tudo que for tratado ou suspeito
* Armazenar os dados limpos no banco de dados
* Fornecer dashboard interativo para acompanhamento

## 2. Problema

Empresas lidam diariamente com arquivos CSV, Excel ou APIs com dados mal formatados, com campos vazios, datas erradas, separadores inconsistentes, entre outros problemas. Essas falhas impactam diretamente na:

* Qualidade de relatórios BI
* Validade de modelos de Machine Learning
* Decisões baseadas em dados incorretos
* Tempo perdido com tratamento manual de dados

## 3. Objetivo Geral

Construir um pipeline ETL com capacidade de **identificar, corrigir e alertar automaticamente** sobre erros, inconsistências e anomalias nos dados, utilizando **Inteligência Artificial**, com foco em dados tabulares.

## 4. Solução Proposta

O pipeline desenvolvido é composto por módulos automatizados, que realizam:

1. **Extração** de dados de fontes como CSV ou APIs
2. **Transformação e tratamento inteligente** com IA
   * Correção com base no padrão dominante
   * Normalização de formatos (datas, números, maiúsculas/minúsculas)
   * Imputação de valores nulos
   * Detecção de outliers/anomalias
3. **Carga** no banco de dados (PostgreSQL)
4. **Geração de log detalhado** por campo com motivo da correção
5. **Envio de relatório por e-mail** com o que foi tratado
6. **Dashboard interativo** com os registros e anomalias encontradas

## 5. Funcionalidades

✅ Correção automática com base na maioria dos registros

✅ Detecção e tratamento de:

* Dados nulos
* Formatos numéricos diferentes (ponto x vírgula)
* Datas fora do padrão
* Campos com maiúsculas/minúsculas inconsistentes
* Outliers com base estatística (z-score ou IQR)

✅ Nenhum campo auxiliar é criado na tabela final

✅ Todos os tratamentos geram:

* Log por registro e por campo
* Motivo da correção
* Indica se houve alteração

✅ Envio automático de relatório por e-mail com:

* Registros tratados
* Campos modificados
* Motivos das correções
* Casos suspeitos (anomalias)

✅ Armazenamento dos dados tratados no PostgreSQL

✅ Dashboard Streamlit com:

* Visualização dos dados limpos
* Filtro por campo corrigido
* Tabela de anomalias encontradas

✅ Pipeline modular e extensível

* Pode receber CSV ou API
* Pode ser agendado (cron)
* Pode ser reusado para outras tabelas

## 6. Stack Tecnológica

* **n8n (via Docker)**: para automação e orquestração dos steps com controle total e sem limitações
* **Python**: para tratamento inteligente dos dados
* **Pandas + Scikit-Learn**: para detecção de anomalias e aplicação de padrões
* **PostgreSQL (local)**: banco relacional local para máxima performance e independência de rede
* **Streamlit**: dashboard para consulta e visualização, construído em Python, demonstrando domínio técnico
* **GitHub**: versionamento do projeto

## 7. Justificativa da Arquitetura

Durante a construção do projeto, optou-se por uma **arquitetura local baseada em Docker** por apresentar:

* Maior controle sobre o ambiente e os dados
* Independência de plataformas externas que impõem limites (como Supabase ou n8n Cloud)
* Capacidade de executar scripts de IA e Python sem restrições
* Demonstração de conhecimentos em infraestrutura, automação e integração

Da mesma forma, a escolha pelo **Streamlit ao invés de ferramentas no-code** (como Loveble) foi feita porque:

* Permite controle total da interface, filtros, layout e regras
* Usa Python puro, provando domínio técnico real
* Evita a imagem de um trabalho “arrastado” ou “pronto”

Estas decisões foram tomadas visando não só um resultado funcional, mas também uma apresentação que **comprove domínio técnico, autonomia de desenvolvimento e aplicação prática de ferramentas de mercado.**

## 8. Benefícios da Solução

* **Redução do retrabalho humano** em tratar dados
* **Ganho de confiabilidade** nos relatórios
* **Correção padronizada e consistente**
* **Automatização com rastreabilidade** (logs e relatórios)
* **Pronto para uso em qualquer empresa com fontes despadronizadas**

## 9. Casos de Uso

* Empresas que recebem dados de vários fornecedores
* Organizações com problemas recorrentes em relatórios
* Times de BI e Engenharia de Dados que precisam validar qualidade
* Situações com integração de fontes diversas (CSV, Excel, APIs)

## 10. Próximos Passos

* Finalizar a parte de tratamento **dinâmico de todos os campos**
* Implementar **envio de e-mail com o log de correções**
* Criar **dashboard interativo no Streamlit**
* Documentar todos os códigos e fluxos
* Validar com a orientadora e preparar para apresentação

## 11. Utilidade do Dashboard e Público-Alvo

O dashboard desenvolvido com Streamlit é o principal meio de \*\*visualizar e auditar o resultado da limpeza \*\***automática feita pela IA**\*\*.

Ele oferece:

* Resumo dos dados tratados
* Visualização de campos com mais correções
* Tabela interativa com filtros por campo
* Gráficos de anomalias e log de correções

### Para quem é útil:

| Público | Como o dashboard ajuda |
| --- | --- |
| Analistas de dados / BI | Validação da qualidade e confiabilidade dos dados |
| Equipe de QA / Governança | Auditoria das decisões automatizadas do pipeline |
| Desenvolvedores / Engenheiros | Validação técnica da automação antes de produção |
| Professores / Banca do TCC | Comprovar resultados de forma visual e prática |

A vantagem do dashboard é **transformar o pipeline invisível em um sistema transparente**, auditável e confiável para qualquer parte interessada.

## 12. Fonte de Dados e Tipos de Tratamento

Para fins de testes e demonstração, o pipeline pode receber dados de:

* Arquivo CSV hospedado no GitHub
* APIs públicas simuladas, como:
  + https://jsonplaceholder.typicode.com/users
  + https://randomuser.me/api/
  + https://fakerapi.it/

Os dados de teste devem conter erros simulados reais, como:

| Campo | Tipo | Problemas simulados |
| --- | --- | --- |
| nome | texto | Maiúsculas/minúsculas, espaços extras |
| email | texto | Emails inválidos, campos nulos |
| nascimento | data | Datas fora do padrão, nulas ou impossíveis |
| preco | número | Vírgula x ponto, negativos, string |
| ativo | booleano | Textos soltos: “sim”, “1”, “verdadeiro” |

Esses dados permitirão a demonstração completa de **tratamento automático + log + visualização no dashboard**.

## 13. Conclusão

Este projeto demonstra que é possível automatizar um pipeline de qualidade de dados com o uso de ferramentas gratuitas e acessíveis, gerando alto valor para qualquer equipe que trabalhe com dados. Além disso, mostra como aplicar IA de forma prática, funcional e responsiva, sem depender de soluções prontas.

Projeto desenvolvido por **Thaissa**, estudante de Sistemas de Informação.  
Orientação e mentoria: Especialista Sênior em Engenharia de Dados.