**Projeto de Automação de Processamento e Carga de Dados - Documentação Técnica**

**1. Introdução**

Este documento tem como objetivo descrever de forma detalhada o processo de extração, transformação e carga de dados (ETL) automatizado, desenvolvido para Rama Advogados Associados. O projeto envolve a leitura de dados de um arquivo Excel, a comparação desses dados com registros já existentes em um banco de dados PostgreSQL, e a inserção de novos registros no banco, utilizando containers Docker para gerenciamento de ambiente. Este documento segue as normas de formatação da ABNT (Associação Brasileira de Normas Técnicas).

**2. Objetivo do Projeto**

O objetivo principal do projeto é automatizar a rotina de atualização dos registros de distribuição de dossiês no banco de dados da empresa. O processo visa garantir que novos registros sejam inseridos de forma eficiente, minimizando a possibilidade de duplicação de dados e reduzindo o esforço manual necessário para a atualização dos registros.

**3. Descrição Geral do Processo ETL**

O processo ETL (Extração, Transformação e Carga) implementado neste projeto pode ser descrito em três etapas principais:

1. **Extração:** Leitura dos dados de um arquivo Excel armazenado em um servidor da empresa.
2. **Transformação:** Processamento dos dados extraídos, incluindo a renomeação de colunas, conversão de datas, e eliminação de registros nulos.
3. **Carga:** Inserção dos novos registros no banco de dados PostgreSQL, utilizando um container Docker para execução dos comandos de inserção.

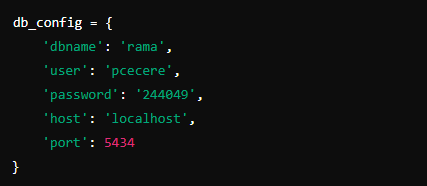
**4. Tecnologias Utilizadas**

* **Python:** Linguagem de programação principal utilizada para a automação do processo.
* **Pandas:** Biblioteca utilizada para a manipulação e transformação dos dados.
* **Docker:** Tecnologia de containers utilizada para isolar o ambiente de execução e facilitar a integração com o banco de dados PostgreSQL.
* **PostgreSQL:** Sistema de gerenciamento de banco de dados relacional onde os registros de distribuição são armazenados.
* **SQLAlchemy:** Biblioteca para interações com o banco de dados PostgreSQL.
* **tarfile:** Biblioteca Python utilizada para manipular arquivos TAR, necessária para a transferência de arquivos para o container Docker.
* **BytesIO:** Utilizado para manipulação de streams de dados em memória.

**5. Descrição Detalhada do Código**

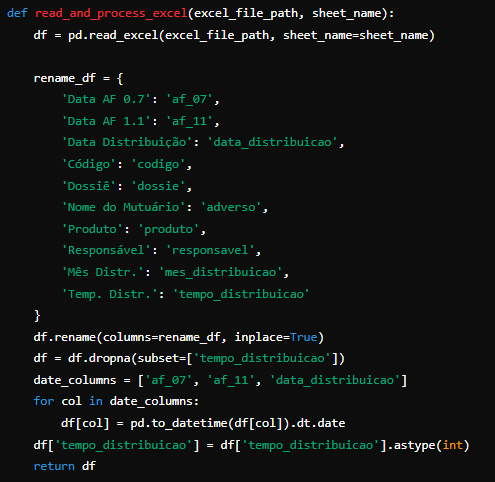
**5.1 Configurações do Banco de Dados**

A conexão com o banco de dados é estabelecida através de um dicionário de configuração (db\_config) que contém as credenciais necessárias para acessar o banco PostgreSQL.

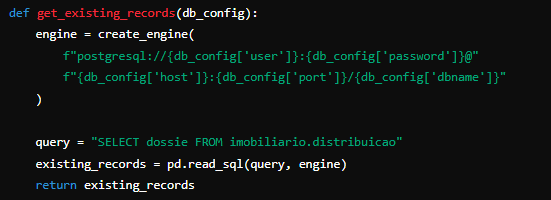


**5.2 Leitura e Processamento dos Dados do Excel**

A função read\_and\_process\_excel é responsável pela leitura do arquivo Excel e pela aplicação das transformações necessárias aos dados. O nome da aba a ser lida é especificado pelo parâmetro sheet\_name.

* **Renomeação de Colunas:** Um dicionário de mapeamento (rename\_df) é utilizado para renomear as colunas do DataFrame, padronizando os nomes para facilitar o uso posterior.
* **Eliminação de Nulos:** Registros sem valores na coluna tempo\_distribuicao são eliminados para garantir a integridade dos dados.
* **Conversão de Datas:** Colunas que representam datas são convertidas para o formato apropriado, utilizando a função pd.to\_datetime.
* **5.3 Consulta aos Registros Existentes**

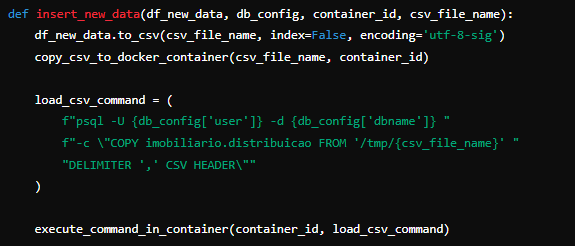
A função get\_existing\_records realiza uma consulta ao banco de dados para obter os dossiês já registrados. Esta etapa é crucial para identificar quais registros no arquivo Excel ainda não foram inseridos no banco.



**5.4 Inserção de Novos Dados**

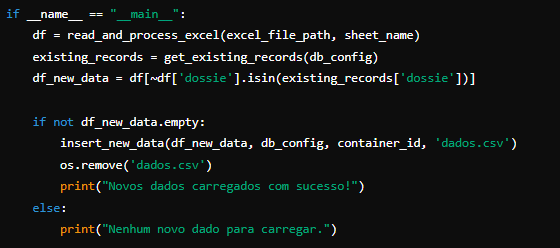
Novos registros, identificados pela ausência de correspondência na coluna ‘dossie’, são salvos em um arquivo CSV, que é então copiado para o container Docker responsável pela execução do banco de dados.

* **Salvamento como CSV:** A função save\_df\_to\_csv é utilizada para salvar o DataFrame em um arquivo CSV.
* **Cópia para o Container Docker:** O arquivo CSV é copiado para o container Docker utilizando a função copy\_csv\_to\_docker\_container.
* **Execução do Comando de Carga no Container:** Após a cópia do arquivo, um comando SQL é executado dentro do container para carregar os dados no banco PostgreSQL.



**5.5 Execução do Processo**

O processo completo é executado na função main, que coordena todas as etapas descritas acima. Se novos registros forem identificados, o processo de inserção é acionado, e uma mensagem de confirmação é exibida.



**6. Decisões Tomadas**

1. **Uso de Docker:** A decisão de utilizar Docker para isolar o ambiente de execução foi tomada para garantir que as dependências do banco de dados e do ambiente de desenvolvimento fossem controladas e reproduzíveis. Além disso, o uso de Docker facilita a integração contínua e a automação de tarefas.
2. **Processamento de Datas:** A conversão de colunas de data foi necessária para padronizar os formatos e facilitar comparações e cálculos no banco de dados.
3. **Comparação de Dossiês:** A opção por comparar os dossiês antes da inserção de novos dados foi tomada para evitar duplicações no banco de dados, assegurando a integridade dos dados.
4. **Manuseio de Arquivos TAR:** Para copiar arquivos para dentro do container Docker, foi necessário utilizar a manipulação de arquivos TAR, devido à forma como o Docker lida com operações de arquivos em containers.

**7. Conclusão**

Este projeto implementa uma solução robusta e escalável para a atualização automática de registros no banco de dados, minimizando o trabalho manual e reduzindo a possibilidade de erros. A utilização de containers Docker, aliada a uma abordagem estruturada de ETL, garante a confiabilidade e a facilidade de manutenção do processo.