Documentação Técnica do Pipeline de Inserção de Dados

1. Visão Geral e Objetivo

Este projeto implementa o processo de Extração, Transformação e Carga (ETL) necessário para preparar e inserir novos dados no ambiente de destino (`d_cnpj`).

O pipeline foi desenvolvido com ênfase máxima em modularidade e documentação, conforme solicitado, visando facilidade de manutenção e validação por Levy.

2. Arquitetura Modular (Para Levy!)

O código está segregado em dois módulos principais, garantindo a separação clara de responsabilidades:

Arquivo	Responsabilidade	Descrição
`main.py`	**Orquestrador / Ponto de Entrada**	Define os parâmetros de ambiente (caminhos, destino) e **executa a pipeline**. É o script que deve ser agendado.
`data_processor.py`	**Lógica de ETL (Core)**	Contém a classe `ETLProcessor` com os métodos isolados: `extract_data()`, `transform_data()`, e `load_data()`.

Vantagem: Se houver necessidade de mudar a regra de transformação, basta alterar *apenas* o método `transform_data` em `data_processor.py`, sem tocar no orquestrador.

3. Fluxo do Processo (`main.py`)

1. **Início:** O script `main.py` é executado.
2. **Inicialização:** Instancia a classe `ETLProcessor` com os caminhos de origem e destino.
3. **`run_pipeline()`:** O método é chamado e sequência as etapas:
    - **E**xtract: Lê o CSV de origem.
    - **T**ransform: Aplica regras de limpeza e padronização (Ex: renomeia colunas, trata nulos).
    - **L**oad: Simula a inserção dos dados limpos no `PROD_DW_VENDAS`.

4. **Logging:** Todas as etapas, advertências (`WARNING`) e erros (`ERROR`) são registrados via módulo `logging`.

4. Detalhes da Transformação (Importante para Validação)

As seguintes regras de negócio e limpeza foram implementadas no método `transform_data()`:

Regra de Transformação	Ação Executada
Padronização de Colunas	Todos os nomes de colunas são convertidos para `snake_case` (minúsculas com `_`).
Tratamento de Nulos (Nulls)	Valores `None` ou `NaN` na coluna 'Status' (e outras textuais) são preenchidos com a string 'N/A'.
Cálculos/Features	Criação da coluna `preco_total` (se necessário), ou outra feature de validação. *(Implementação básica para demonstração; expandir conforme a necessidade real).*

5. Próximos Passos

A carga final (o *L* do ETL) está simulada no `load_data()`

Para colocar em produção, é preciso substituir a simulação de carga:

```
# data_processor.py (trecho do método load_data)

# --- SIMULAÇÃO DA INSERÇÃO REAL ---
# Esta seção deve ser substituída pelo código de conexão real.
logging.info(f"SUCESSO: Simulação de inserção de {len(df)} registros...")
# ---------------------------------

# Exemplo de código real (Ajustar conforme o SGBD)
# engine = create_engine('postgresql://user:pass@host/db')
# df.to_sql('tabela_destino', engine, if_exists='append', index=False)
```

6. Como Executar

1. pip install pandas

Execução:

2. python main.py

3. **Monitoramento:** Acompanhe o console para verificar o log de `INFO` e confirmar o status de sucesso ou falha.