📘 Documentação do Software

Pipeline ETL – IQVIA

---

1. Introdução

Este software implementa um **pipeline ETL (Extract, Transform, Load)** para processamento de dados da base IQVIA, realizando:

- Extração de dados de arquivos fonte (IQVIA e Filial-Brick);
- Transformação, padronização e enriquecimento dos dados;
- Carga em um **Data Warehouse PostgreSQL**, estruturado em **dimensões** e **fato**.

O objetivo principal é permitir análises de mercado, participação e volume de vendas por *brick* e *filial*.

---

2. Arquitetura do Sistema

O projeto segue uma arquitetura modular, organizada da seguinte forma:

```
M1S10-main/
│
├── main.py
├── src/
│   ├── database.py
│   ├── extract.py
│   └── etl/
│       ├── pipeline.py
│       ├── transform.py
│       └── load.py
│
├── data/
│   ├── iqvia.csv
│   └── filial_brick.csv
│
└── tests/
    └── test_sanity.py
```

---

3. Tecnologias Utilizadas

- **Python 3.13**
- **Pandas** – manipulação e transformação de dados
- **psycopg2** – conexão com PostgreSQL
- **PostgreSQL** – banco de dados relacional
- **pgAdmin** – administração e visualização dos dados

---

4. Modelo de Dados (Data Warehouse)

4.1 Dimensão Brick – `dw.dim_brick`

Campo	Tipo	Descrição
id_brick	SERIAL	Chave primária
brick_nome	VARCHAR	Nome do brick

4.2 Dimensão Filial – `dw.dim_filial`

Campo	Tipo	Descrição
id_filial	SERIAL	Chave primária
cod_filial	INTEGER	Código da filial
id_brick	INTEGER	FK para dim_brick

4.3 Fato Vendas – `dw.fact_vendas`

Campo	Tipo	Descrição
id_venda	SERIAL	Chave primária
id_brick	INTEGER	FK para dim_brick
ean	BIGINT	Código do produto
periodo	VARCHAR	Período da venda

vol_pp	NUMERIC	Volume PP
vol_concorrente_indep	NUMERIC	Volume concorrente
vol_total_mercado	NUMERIC	Volume total
participacao_clamed	NUMERIC	Market share

---

5. Funcionamento do Pipeline ETL

5.1 Execução Principal

O pipeline é iniciado pelo arquivo `main.py`:

from src.etl.pipeline import run_pipeline

run_pipeline()

5.2 Extract

Arquivo: `src/extract.py`

Responsável por:

- Ler arquivos CSV
- Retornar DataFrames Pandas

Funções principais:

- `extract_iqvia()`
- `extract_filial_brick()`

5.3 Transform

Arquivo: `src/etl/transform.py`

Principais responsabilidades:

- Normalização de nomes de colunas (snake_case)
- Conversão de tipos
- Criação de métricas calculadas

- Enriquecimento com IDs do banco

Funções principais:

- `transform_iqvia(df)`
- `transform_filial_brick(df)`
- `enrich_iqvia_with_ids(df, conn)`

5.4 Load

Arquivo: `src/etl/load.py`

Responsável por inserir dados no banco utilizando **itertuples()**, garantindo:

- Melhor performance
- Menos erros de chave
- Código mais legível

Carga realizada em três etapas:

1. `load_dim_brick`
2. `load_dim_filial`
3. `load_fact_vendas`

---

6. Conexão com Banco de Dados

Arquivo: `src/database.py`

```
def connect_db():
    return psycopg2.connect(
        dbname="etl_Iquivia",
        user="postgres",
        password="******",
        host="localhost",
        port=5432
    )
```

---

7. Execução do Projeto

7.1 Pré-requisitos

- PostgreSQL instalado
- Banco e schema `dw` criados
- Tabelas do Data Warehouse criadas
- Dependências Python instaladas

pip install pandas psycopg2

7.2 Executar o Pipeline

python main.py

---

8. Validação dos Dados

No pgAdmin, os dados podem ser visualizados com:

```
SELECT * FROM dw.dim_brick;
SELECT * FROM dw.dim_filial;
SELECT * FROM dw.fact_vendas;
```

---

9. Testes

Arquivo: `tests/test.py`

Testes básicos garantem:

- Conexão com banco
- Leitura dos dados
- Estrutura mínima dos DataFrames

---

10. Considerações Finais

Este software foi desenvolvido com foco em:

- Organização
- Robustez
- Facilidade de manutenção

O pipeline é escalável e pode ser adaptado para novas fontes de dados ou métricas analíticas.