**Documentação Técnica Projeto de ETL - Fórmula 1   
Autor: Alyff Antonio   
Data: 06/06/2025  
  
  
Documentação: Projeto de ETL de Dados da Fórmula 1**

**1. Visão Geral**

Este projeto implementa um pipeline de **ETL (Extração, Transformação e Carga)** para processar dados históricos de corridas de Fórmula 1. O sistema é projetado para ser robusto e de fácil execução, utilizando Python para a lógica de manipulação de dados e Docker para criar um ambiente de desenvolvimento e produção consistente e isolado.

Os dados são extraídos de arquivos CSV, passam por um processo de transformação e limpeza com a biblioteca pandas, e são carregados em um banco de dados PostgreSQL, onde são organizados em tabelas e views para facilitar consultas e análises.

**2. Funcionalidades Principais**

* **Extração (Extract):** Leitura de dados de quatro arquivos CSV distintos:
  + constructors.csv: Dados das equipes.
  + drivers.csv: Dados dos pilotos.
  + races.csv: Dados das corridas.
  + results.csv: Resultados de cada corrida.
* **Transformação (Transform):** Aplicação de tratamentos e limpezas nos dados, como a criação de "slugs" (identificadores únicos amigáveis) para os nomes dos pilotos.
* **Carga (Load):** Inserção dos dados processados em um banco de dados PostgreSQL.
* **Setup do Banco de Dados:** Criação automática de tabelas e views SQL através de scripts para permitir análises de performance, como:
  + v\_driver\_race\_performance: Visão de performance dos pilotos.
  + v\_fastest\_lap\_per\_race: Visão das voltas mais rápidas.
* **Ambiente Containerizado:** Uso de Docker e Docker Compose para gerenciar a aplicação e o banco de dados, simplificando a configuração do ambiente.

**3. Tecnologias Utilizadas**

* **Linguagem de Programação**: Python 3.12
* **Banco de Dados**: PostgreSQL
* **Containerização**: Docker e Docker Compose
* **Bibliotecas Python**:
  + pandas: Essencial para a manipulação e análise dos dados.
  + SQLAlchemy: ORM para a interação com o banco de dados PostgreSQL.
  + psycopg2-binary: Driver de conexão para o PostgreSQL.
  + python-dotenv: Gerenciamento de variáveis de ambiente.
  + python-slugify: Criação de slugs a partir de strings.
  + requests, beautifulsoup4, lxml: Incluídas para futuras expansões (ex: web scraping de dados).

**4. Estrutura do Projeto**

/

├── docker-compose.yml # Orquestra os contêineres da aplicação e do banco de dados.

├── extraction/ # Contém os arquivos CSV de origem.

│ ├── constructors.csv

│ ├── drivers.csv

│ ├── races.csv

│ └── results.csv

├── main.py # Ponto de entrada que executa todo o pipeline ETL.

├── requirements.txt # Lista de dependências Python.

├── scripts\_db/ # Scripts para criação do schema e views no banco.

│ ├── create\_table.sql

│ ├── create\_views.sql

│ ├── query\_fast\_lap.sql

│ └── query\_race\_perform.sql

└── src/ # Código-fonte da aplicação.

├── extraction.py # Módulo de extração de dados.

├── loading.py # Módulo de carga de dados para o banco.

├── setup\_db.py # Módulo de configuração do banco de dados.

└── transformation.py # Módulo de transformação dos dados.

**5. Como Executar o Projeto**

**5.1. Pré-requisitos**

* **Git**: Para clonar o repositório.
* **Docker**: Para execução dos contêineres.
* **Docker Compose**: Para orquestração dos serviços definidos no docker-compose.yml.

**5.2. Passos para Execução**

1. **Clone o repositório** para sua máquina local.
2. **Variáveis de Ambiente**: O docker-compose.yml já está configurado com as variáveis de ambiente necessárias para a conexão entre a aplicação e o banco de dados. Nenhuma configuração adicional é necessária para a execução padrão.
3. **Inicie os Contêineres**: Abra um terminal na pasta raiz do projeto e execute o comando abaixo. Ele irá construir as imagens e iniciar os contêineres em segundo plano.

Bash

docker-compose up --build -d

1. **Verifique a Execução**: O script main.py é executado automaticamente quando o contêiner app é iniciado. Ele realizará todo o processo de ETL. Para acompanhar os logs da aplicação e verificar se tudo ocorreu bem, use o comando:

Bash

docker-compose logs -f app

Ao final da execução, os dados estarão carregados no banco de dados PostgreSQL.

**6. Acesso e Análise dos Dados**

Após a execução bem-sucedida, você pode se conectar ao banco de dados PostgreSQL usando qualquer cliente de banco de dados (como DBeaver, DataGrip ou pgAdmin).

* **Host**: localhost #com Docker local.docker.internal
* **Porta**: 5432
* **Usuário**: postgres
* **Senha**: postgres
* **Banco de Dados**: formula\_db

Dentro do banco, você encontrará as tabelas e views criadas pelos scripts. Para analisar os dados, você pode executar as queries disponíveis na pasta scripts\_db/.