Documentação do Desafio da Raízen - ANP

Introdução

Esta documentação fornece uma visão geral da DAG, tarefas, dependências, configuração do banco de dados e detalhes sobre os processos de transformação e carregamento de dados. O objetivo é orientar na compreensão e execução do código fornecido.

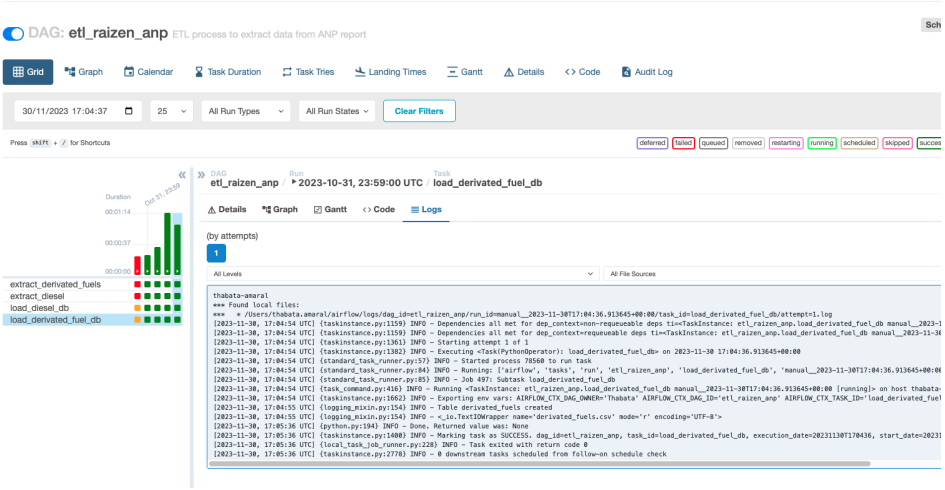
Setup

- Airflow: localhost - Porta 8029.
- PostgreSQL
- Conversão do arquivo 'xls' para 'xlsx.'

Características da Dag:

- ID: `etl_raizen_anp`
- Descrição: orquestra o processo ETL para extrair, transformar e carregar dados de um relatório da ANP.
- Data de Início: 29 de novembro de 2023
- Intervalo: será executado às 23h59 no último dia de cada mês.
- Argumentos Padrão:
    - Proprietário: Thabata
    - Retentativas: 3 (configuradas para tentar novamente tarefas falhadas até 3 vezes)

A imagem abaixo mostra o sucesso na execução da DAG.



Tarefas:

1. `extract_derivated_fuels`:
    - ID da Tarefa: `extract_derivated_fuels`
    - Função Python: `etl_transform`
    - Parâmetros:
        - `sheet_name`: 1 (Índice da planilha no arquivo Excel)
        - `table_name`: 'derivated_fuels'

2. `extract_diesel`:
    - ID da Tarefa: `extract_diesel`
    - Função Python: `etl_transform`
    - Parâmetros:
        - `sheet_name`: 2 (Índice da planilha no arquivo Excel)
        - `table_name`: 'diesel'

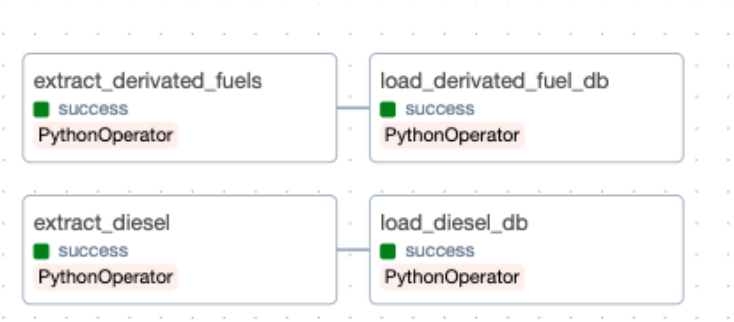
3. `load_diesel_db`:
    - ID da Tarefa: `load_diesel_db`
    - Função Python: `etl_load`
    - Parâmetros:
        - `table_name`: 'diesel'

4. `load_derivated_fuel_db`:
    - ID da Tarefa: `load_derivated_fuel_db`
    - Função Python: `etl_load`
    - Parâmetros:
        - `table_name`: 'derivated_fuels'

Dependências entre Tarefas:

- As tarefas `extract_diesel` e `extract_derivated_fuels` são executadas de forma independente.
- A tarefa `load_diesel_db` depende da conclusão da tarefa `extract_diesel`.
- A tarefa `load_derivated_fuel_db` depende da conclusão da tarefa `extract_derivated_fuels`.

A imagem abaixo mostra a dependência das tarefas.



Configuração do Banco de Dados:

- Nome do Banco de Dados: `postgres`
- Usuário: `postgres`
- Senha: `postgres`
- Host: `localhost`
- Porta: `5432`

Transformação de Dados (função etl_transform):

- Lê os dados de um arquivo Excel localizado em `'../raw_data/vendas-combustiveis-m3.xlsx'`.
- Renomeia colunas e realiza a remodelagem dos dados usando a biblioteca pandas.
- Cria uma coluna 'year_month' combinando as colunas 'Ano' e 'variable'.
- Remove colunas desnecessárias e preenche valores NaN com 0.
- Adiciona colunas 'unit' e 'created_at' ao DataFrame.
- Salva os dados transformados como um arquivo CSV no diretório '../staging'.

As imagens abaixo mostram os .csvs resultantes.

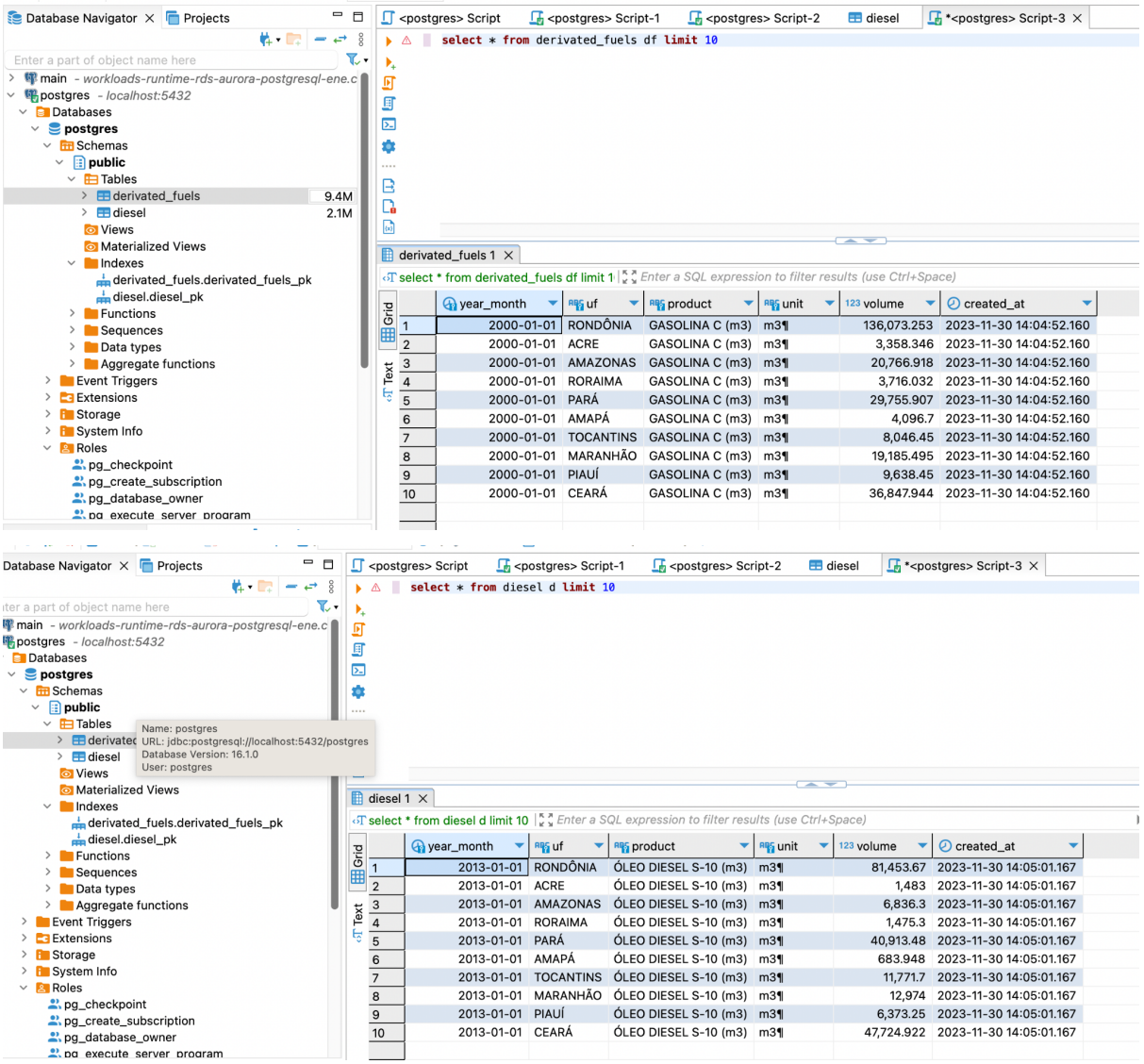
```
product,uf,volume,year_month,unit,created_at
GASOLINA C (m3),RONDÔNIA,136073.253,2000-01-01,m3,2023-11-30 14:04:45.901010
GASOLINA C (m3),ACRE,3358.346,2000-01-01,m3,2023-11-30 14:04:45.901010
GASOLINA C (m3),AMAZONAS,20766.918,2000-01-01,m3,2023-11-30 14:04:45.901010
GASOLINA C (m3),RORAIMA,3716.032,2000-01-01,m3,2023-11-30 14:04:45.901010
GASOLINA C (m3),PARÁ,29755.907,2000-01-01,m3,2023-11-30 14:04:45.901010
GASOLINA C (m3),AMAPÁ,4096.7,2000-01-01,m3,2023-11-30 14:04:45.901010
GASOLINA C (m3),TOCANTINS,8046.45,2000-01-01,m3,2023-11-30 14:04:45.901010
GASOLINA C (m3),MARANHÃO,19185.495,2000-01-01,m3,2023-11-30 14:04:45.901010
GASOLINA C (m3),PIAUÍ,9638.45,2000-01-01,m3,2023-11-30 14:04:45.901010
```

```
product,uf,volume,year_month,unit,created_at
ÓLEO DIESEL S-10 (m3),RONDÔNIA,81453.67,2013-01-01,m3,2023-11-30 14:04:55.649869
ÓLEO DIESEL S-10 (m3),ACRE,1483.0,2013-01-01,m3,2023-11-30 14:04:55.649869
ÓLEO DIESEL S-10 (m3),AMAZONAS,6836.3,2013-01-01,m3,2023-11-30 14:04:55.649869
ÓLEO DIESEL S-10 (m3),RORAIMA,1475.3,2013-01-01,m3,2023-11-30 14:04:55.649869
ÓLEO DIESEL S-10 (m3),PARÁ,40913.48,2013-01-01,m3,2023-11-30 14:04:55.649869
ÓLEO DIESEL S-10 (m3),AMAPÁ,683.948,2013-01-01,m3,2023-11-30 14:04:55.649869
```

Carregamento de Dados (função etl_load):

- Conecta-se ao banco de dados PostgreSQL usando psycopg2.
- Cria a tabela especificada se ainda não existir.
- Trunca a tabela para remover dados existentes.
- Carrega dados do arquivo CSV para a tabela.
- Escolha de chave primária composta: 'year_month', 'uf', 'product' e 'unit'
- Utiliza a cláusula `ON CONFLICT` para lidar com conflitos na restrição única especificada.

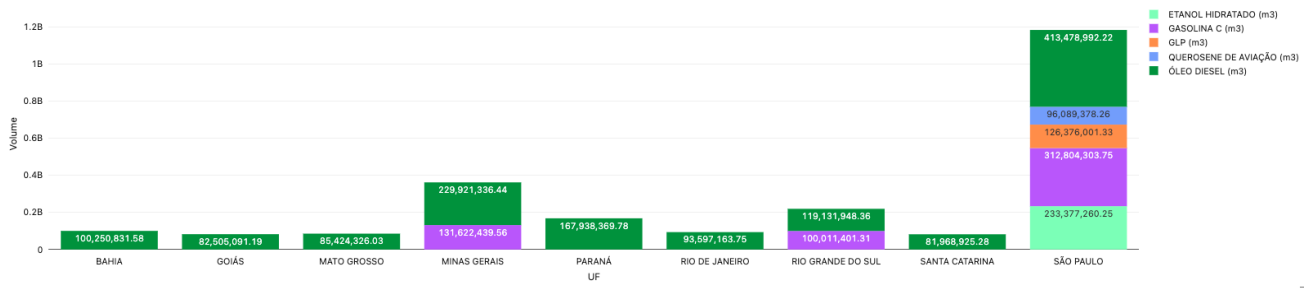
As imagens abaixo mostram as tabelas no PostgreSQL.



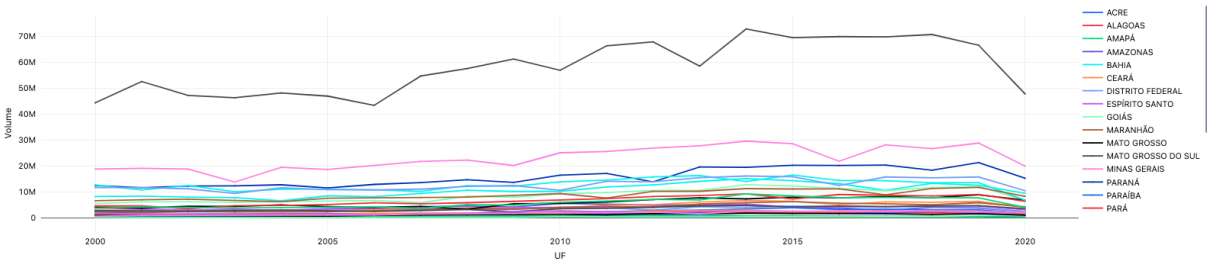
Análise Exploratória Inicial - feita no Databricks:

Análise de produtos derivados de combustíveis de petróleo de 2000 a 2020:

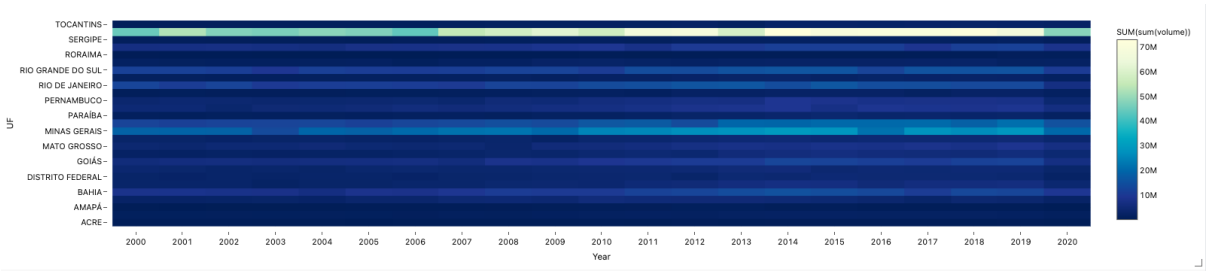
- A produção de diesel corresponde a 41,8% da produção total .
- 4671 registros com volume zero (~8,6% do total de registros).
- Top 15 UF com maior volume produzido, separado por produto.



- São Paulo sempre (2000-2020) foi a UF com maior volume produzido.



- Heatmap mostrando o volume produzido por UF por ano. Quanto mais claro, maior a produção.



Observações:

- Particionamento: As tabelas não estão particionadas no código fornecido. Para este exemplo optei pela indexação tradicional de chave primária e considerei a volumetria dos dados. Tratando-se de um conjunto de dados maior uma ideia de particionamento simples e performático seria:

```
CREATE TABLE IF NOT EXISTS {table_name} (

    id serial,

    year_month date,

    uf VARCHAR(256),

    product VARCHAR(256),
```

```
    unit VARCHAR(256),

    volume FLOAT,

    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP

)

PARTITION BY RANGE (EXTRACT(YEAR FROM year_month));
```

- Tratamento de Erros: A manipulação de exceções está implementada para capturar e imprimir quaisquer erros durante o processo ETL.