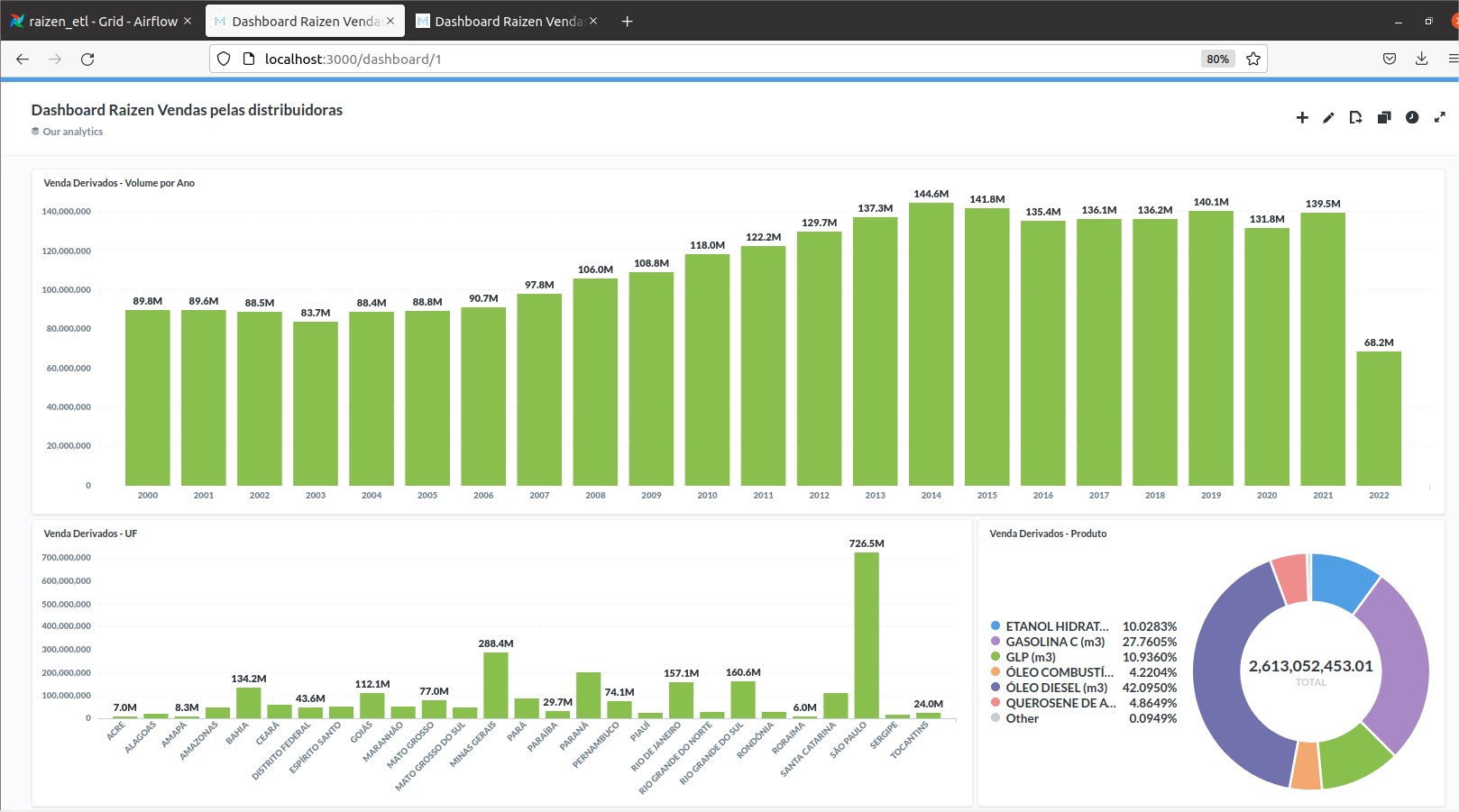
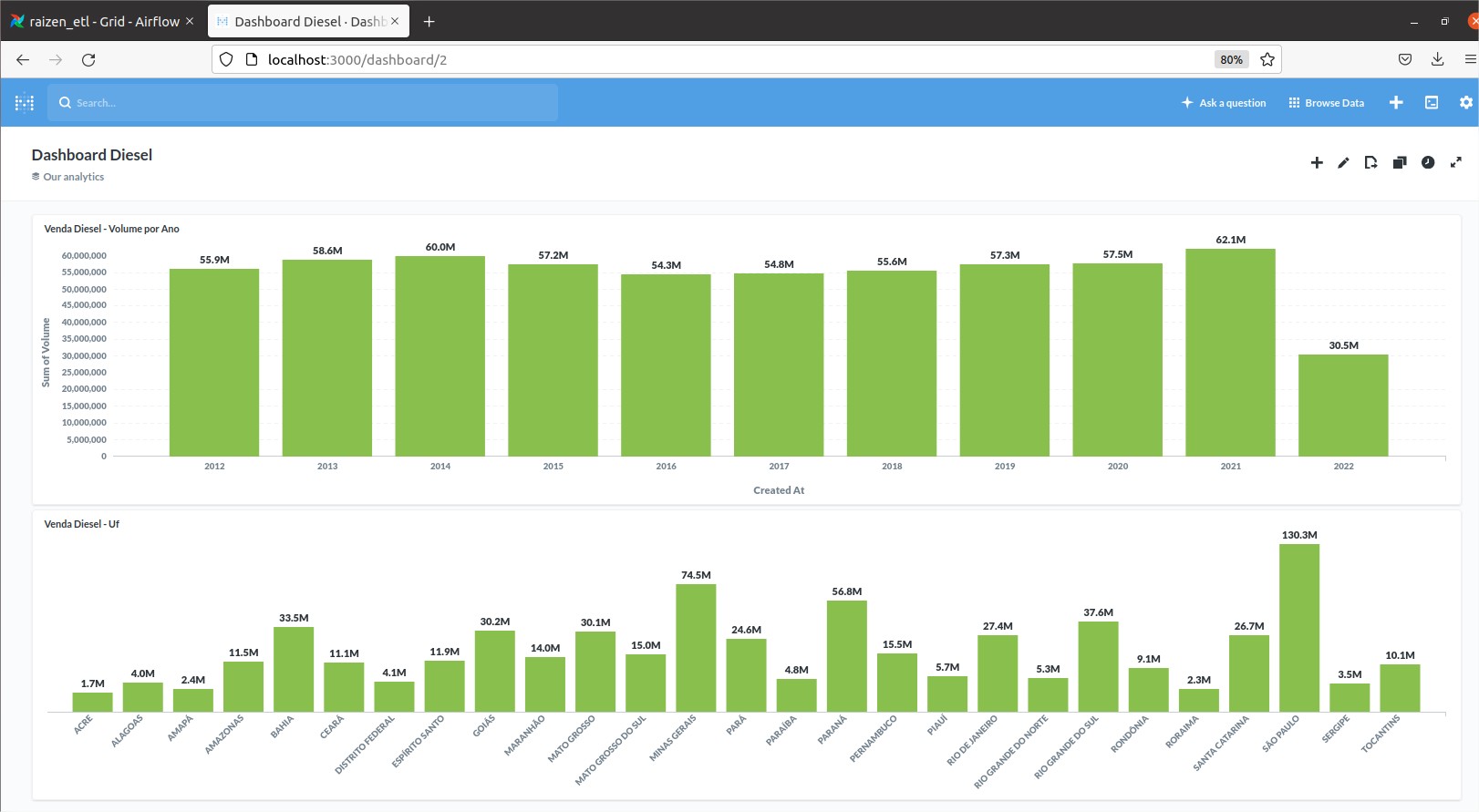
# Documentação Case Raízen

Github:

## Dashboard



Dashboard de Vendas dos derivados de petróleo



Dashboard de Vendas de Diesel

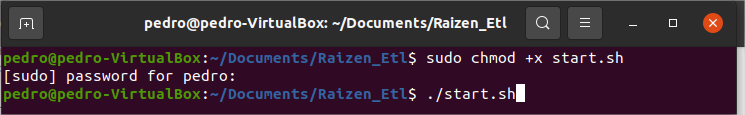
## Iniciando o ambiente

Para iniciar o ambiente é necessário realizar a instalação do docker e do docker-compose na máquina local.

* Documentação para instalação:
  + [Install Docker Engine](https://docs.docker.com/engine/install/)
  + [Install Docker Compose](https://docs.docker.com/compose/install)

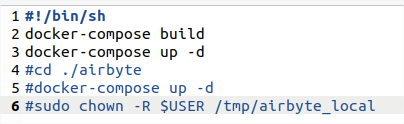
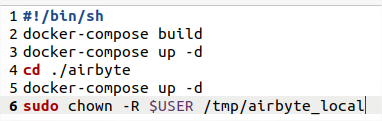
### **Dando start aos containers (sem airbyte)**

Após a instalação para iniciar os containers basta iniciar o script **“start.sh”** antes de realizar o start talvez seja necessário dar a permissão ao arquivo. E após isso inicia-lo conforme a imagem abaixo.

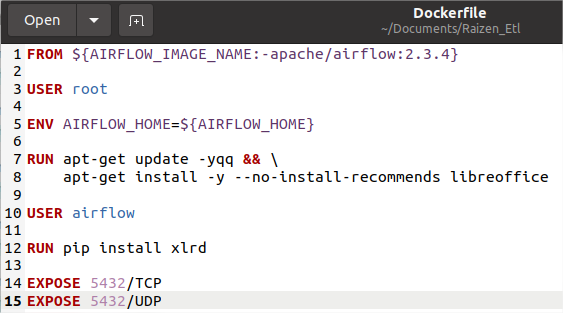


### **Dando start aos containers (com airbyte)**

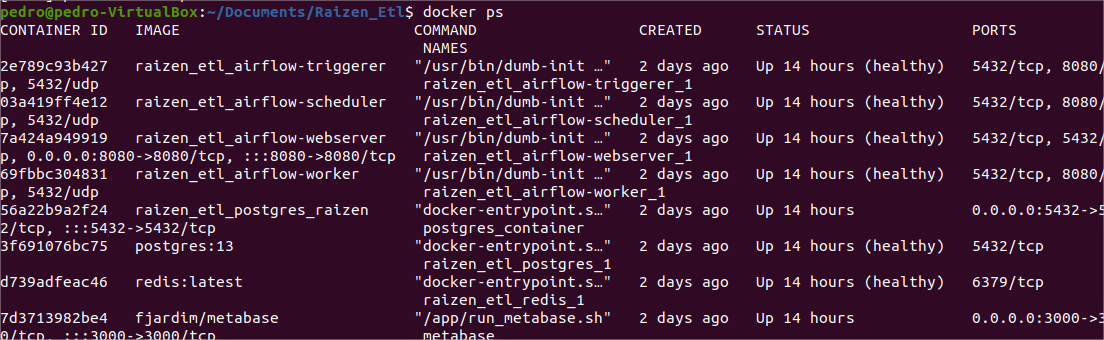
Para iniciar o container com o uso do airbyte para realizar o ELT é necessário editar o arquivo “start.sh”. Para isso abra o arquivo em um bloco de notas e tire o “#” das últimas 3 linhas e salve o arquivo. Após isso no cmd volte a iniciar o script “./start.sh”

Original Editado

Após o start, de forma automática o docker fará o build do docker-compose que é o responsável por fazer algumas instalações e configurações dentro dos containers. Existem 2 arquivos do tipo Dockerfile que fazem a inicialização do container “Airflow” e do container “Postgres”.

* Airflow:
  + O arquivo se encontra na raiz “Dockerfile”
  + 
  + Este docker file instala algumas bibliotecas que são necessárias para o funcionamento do script.

Após o start o seu docker deve estar da seguinte maneira:



Todos os containers estão operacionais e funcionando.

### **Iniciando o Script via Airflow (sem airbyte)**

Para iniciar o script via airflow, basta abrir o navegador e digitar <http://localhost:8080/>

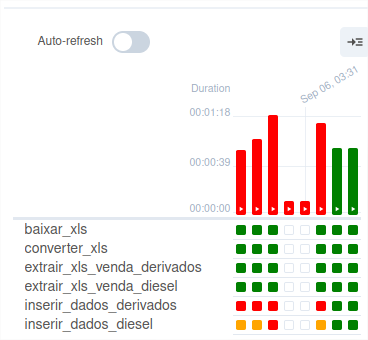
(Caso não apareça nada espere alguns segundos para o airflow iniciar). Após isso faremos o login dentro do airflow:

* User: airflow
* Senha: airflow

Ao entrar dentro do airflow basta buscar a dag “raizen\_etl” e clicar nela.

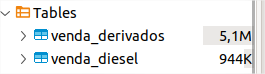


Dentro da dag basta clicar no simbo de “play” para iniciar o script.

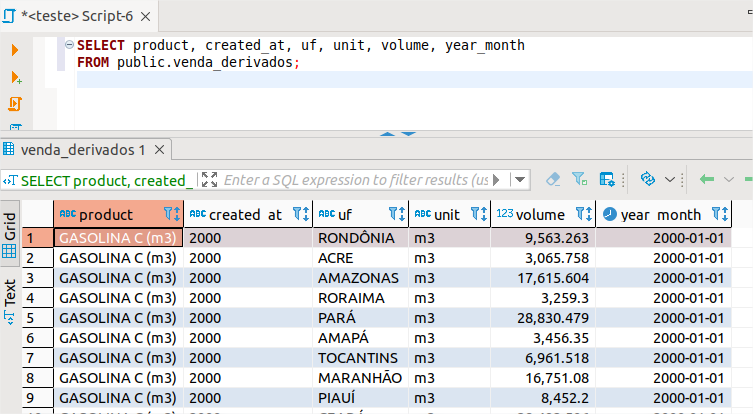


Caso tudo dê certo, todas as tasks ficam verdes.

Após o airflow rodar todas as tasks, os dados já estarão disponíveis dentro do nosso banco de dados postgreeSQL.

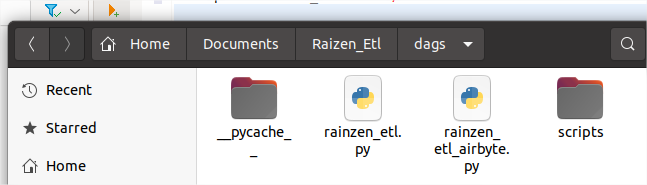


São criadas 2 tabelas, uma com dados das vendas dos derivados de petróleo e outras apenas com as vendas do diesel. (Para acessar ao postgre mostro as conexões mais abaixo):

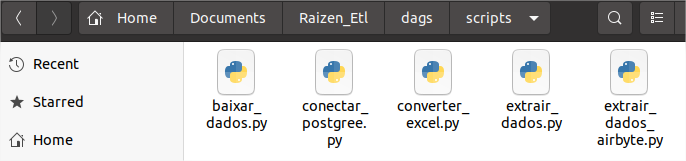


### **Funcionamento do Script**

O script que realiza todo o processo de extração, transformação e ingestão está feito em python. E se encontra dentro da pasta “dags” ./dags/

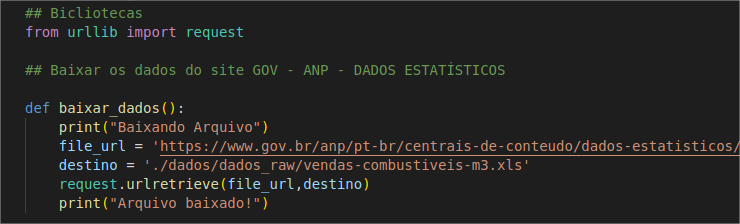


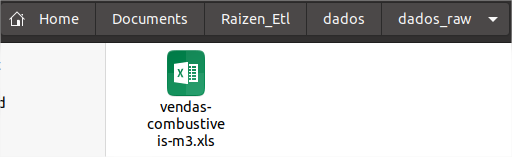
Dentro da pasta dags é temos 2 scripts python e uma pasta com outros scripts:

* **Raizen\_etl**
  + Dag que foi mostrada acima, roda o script com o fluxo completo até a ingestão do banco de dados.
* **Raizen\_etl\_airbyte**
  + Dag que realiza os mesmos procedimentos exceto a ingestão para o banco de dados que fica na responsabilidade do airbyte
* Pasta Scripts
  + Nesta pasta se encontram os módulos para execução.
  + 

#### **Descrição dos arquivos:**

#### **baixar\_dados.py**

Este módulo fica responsável por baixar o excel “vendas-combustiveis-m3.xls” do site do governo [ANP Dados Estatísticos](https://www.gov.br/anp/pt-br/centrais-de-conteudo/dados-estatisticos/) e salvar o arquivo xls dentro da pasta “./dados/dados\_raw/”

O script acima, e após a sua execução o arquivo baixado na pasta dados\_raw

#### **converter\_excel.py**

Este módulo fica responsável por realizar uma transformação do arquivo excel baixado para que seja possível fazer a extração dos seus dados.

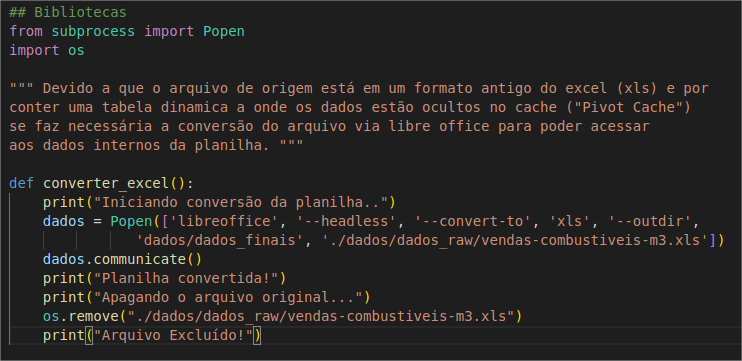
Devido a que o arquivo de origem está em um formato antigo do excel (xls) e por

conter uma tabela dinâmica a onde os dados estão ocultos no cache ("Pivot Cache")

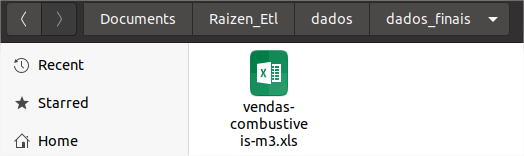
se faz necessária a conversão do arquivo via libre office para poder acessar

aos dados internos da planilha.

Para realizar essa conversão usamos a função **“Popen”** passando os parâmetros do libre office conforme a imagem abaixo.



Após fazer a conversão o dado é salvo na pasta **dados\_finais (./dados/dados\_finais)** e o arquivo original da pasta dados\_raw é apagado.

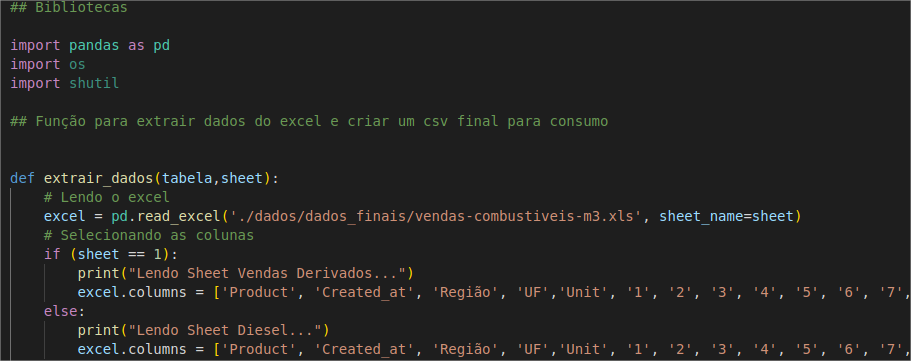


#### **Extrair\_dados.py / Extrair\_dados\_airbyte.py**

Este módulo é o mais longo mas basicamente ele usa o pandas para fazer toda a transformação do arquivo excel e salvar o mesmo como um csv para ingestão posterior do banco de dados.

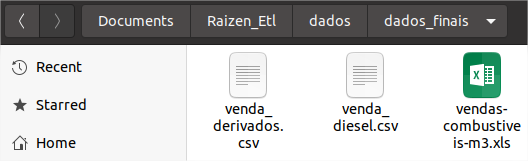
A função gerada se chama **“extrair\_dados(tabela,sheet)**” que deve receber 2 parâmetros de entrada, o primeiro é o nome da tabela e o segundo o número da sheet que será extraída.

Após isso fazemos a leitura do arquivo através da função **pandas.read\_exce**l a onde dentro dela passamos o path do arquivo convertido e a sheet correspondente.



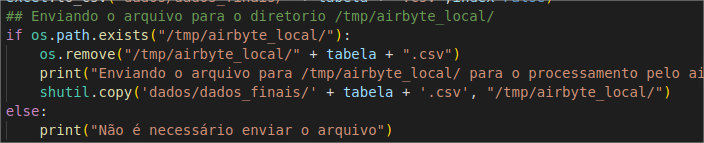
No nosso caso devemos passar a **sheet número 1 que contém os dados das vendas dos derivados e a sheet 3 que contém os dados do diesel**. Como a sheet 1 possui uma coluna a mais, aplicamos um if simples na sheet para que leia de acordo com a sheet correta.

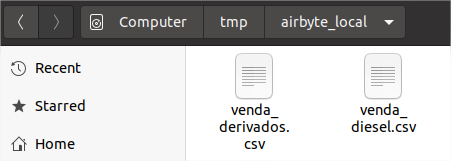
Após todas as transformações salvamos o arquivo no formato .csv dentro da pasta dados finais.



A diferença entre o “extrair\_dados” e o “extrair\_dados\_airbyte” é que no final do código no o script do airbyte verifica se existe a pasta /tmp/airbyte\_local/"

e após isso copie os arquivos csv para essa pasta que é a que o airbyte usa para pegar os arquivos locais.





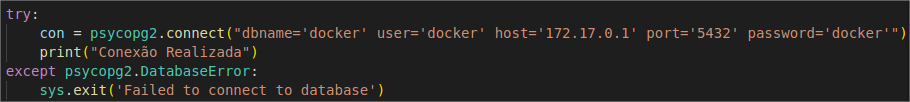
O arquivo start.sh da permissão de acesso através do comando **“sudo chown -R $USER /tmp/airbyte\_local”** se por um acaso acontecer algum erro de acesso negado por parte do airflow ou do script, verifique as pasta dentro da raiz ./tmp/airbyte\_local. Caso apareça uma cadeado execute o comando acima.

**Conectar\_postgree.py**

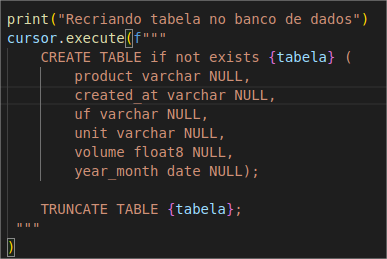
Por último temos o módulo que realiza a conexão com o postgree, cria a tabela final e insere os dados dos arquivos csv via copy.

Dentro do script temos a função **“conect\_postgree(tabela”)** que recebe como parâmetro o nome da tabela que será criada.

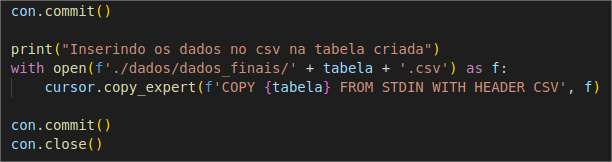
Dentro da função fazemos uma tentativa de conexão com o banco de dados local.



Após isso criamos a tabela com o nome que passamos no parâmetro e fazemos um truncate também para limpar a mesma caso já exista no banco.



Por último inserimos os dados dos arquivos csv que se encontram na pasta **(./dados/dados\_finais/)** através da função copy\_expert do cursor.

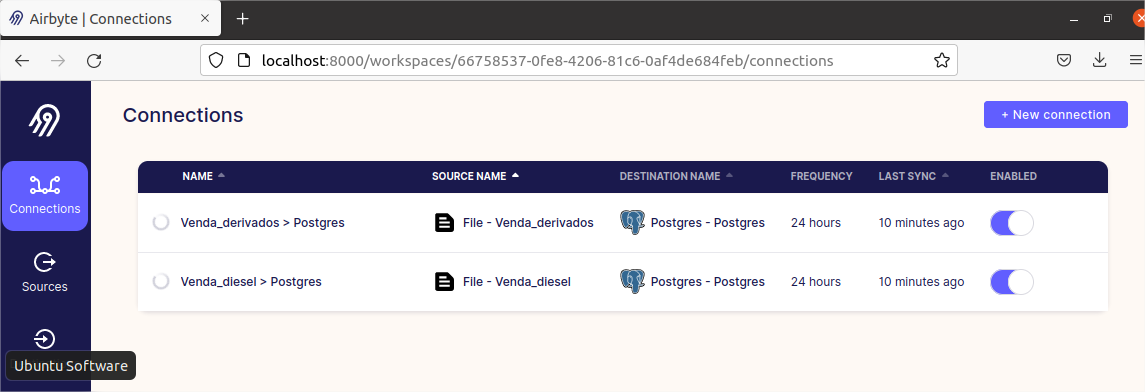


### **Airbyte - Configuração e Funcionamento**

O Airbyte é um projeto de código aberto com diversos conectores, possibilidade de realizar normalização de dados e uma comunidade ativa com mais de quatro mil contribuições.

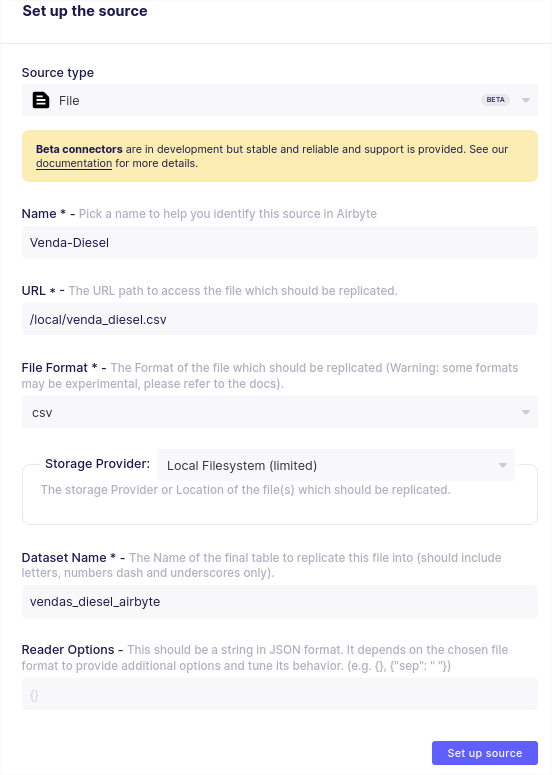
Para saber um pouco mais sobre: <https://medium.com/dados-para-todos/airbyte-integra%C3%A7%C3%A3o-de-dados-cdc-para-equipes-de-dados-modernas-1cc164f895f2>

Caso opte pelo uso do airbyte a sua configuração é muito fácil. Para acessar a interface basta entrar na url -> localhost:8000



Dentro da home podemos ver as conexões existentes.

Para criar uma conexão nova é bem simples, basta clicar em “+ New Connection”



#### **Criando conexão de origem**

1. Dentro da página de nova conexão, precisamos criar um conector de origem, no nosso caso deve ser um conector de arquivo, já que os nossos dados estão no formato .csv. Para isso, escolhemos o **Source Type -> File**
2. Colocamos um nome para o nosso arquivo no caso -> Venda-Diesel
3. Na Url colocamos -> /local/venda\_diesel.csv
   1. A pasta local do airbyte mapeia a pasta ./tmp/airbyte\_local/ do nosso host a onde o script envia os arquivos csv.
4. No File format escolhemos csv.
5. No Storage Provider selecionamos -> Local Filesystem
6. No dataset name colocamos o nome da tabela que será criada -> vendas\_diesel\_airbyte
7. Por último clicamos em “Set up source”
8. Dando tudo certo iremos para a página de criar uma conexão de destino.

#### 

#### **Criando conexão de destino**

1. Na escolha do tipo de source escolhemos o **Source Type -> Postgres**
2. Colocamos o nome da conexão que queremos -> Postgres DB
3. No host colocamos -> localhost
4. No port mantemos o 5432
5. No DB name colocamos -> docker
6. No Schema mantemos o public
7. No user e no password colocamos docker
8. Clicamos em set up destination
9. Dando tudo certo a conexão será criada.

#### **Criando a conexão final**

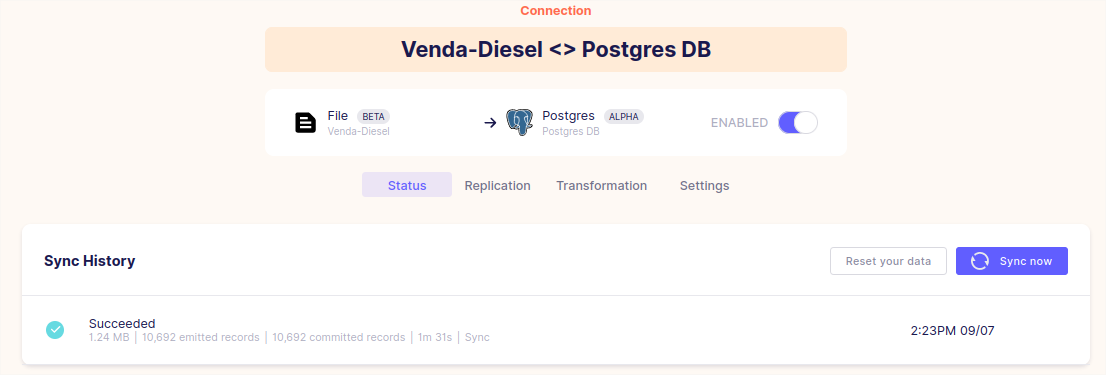
Depois de ter criado a conexão uma nova página será aberta onde precisamos colocar o nome da conexão e configurar outros parâmetros caso seja necessário. No nosso caso basta apenas clicar em “set up connection” para salvar a conexão.

#### **Executando o job**

Após criar a conexão no passo anterior uma nova página vai aparecer com a conexão ativa e rodando pela primeira vez de forma automática.

O airbyte exige muita cpu e ram da máquina por este motivo o deixei como opcional no job. Caso de algum erro tente fechar os aplicativos da sua máquina e rodar novamente.

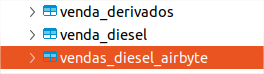
Se a primeira execução funcionar aparecerá a seguinte mensagem.



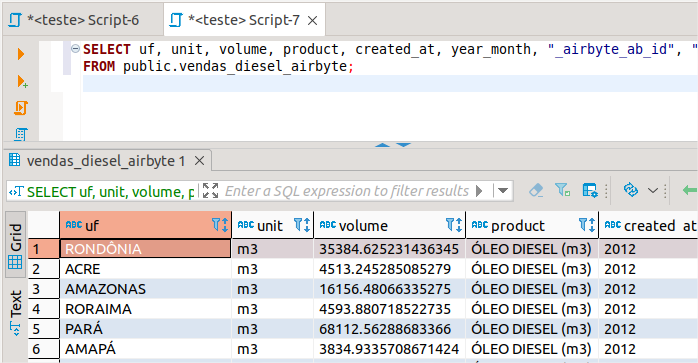
Neste caso o job rodou com sucesso, gravando 10.692 linhas no nosso banco.

#### **Verificando a Tabela no Banco de Dados**

Dando tudo certo será gerada uma tabela com os dados com o nome que demos na configuração de origem no nosso caso (vendas\_diesel\_airbyte).



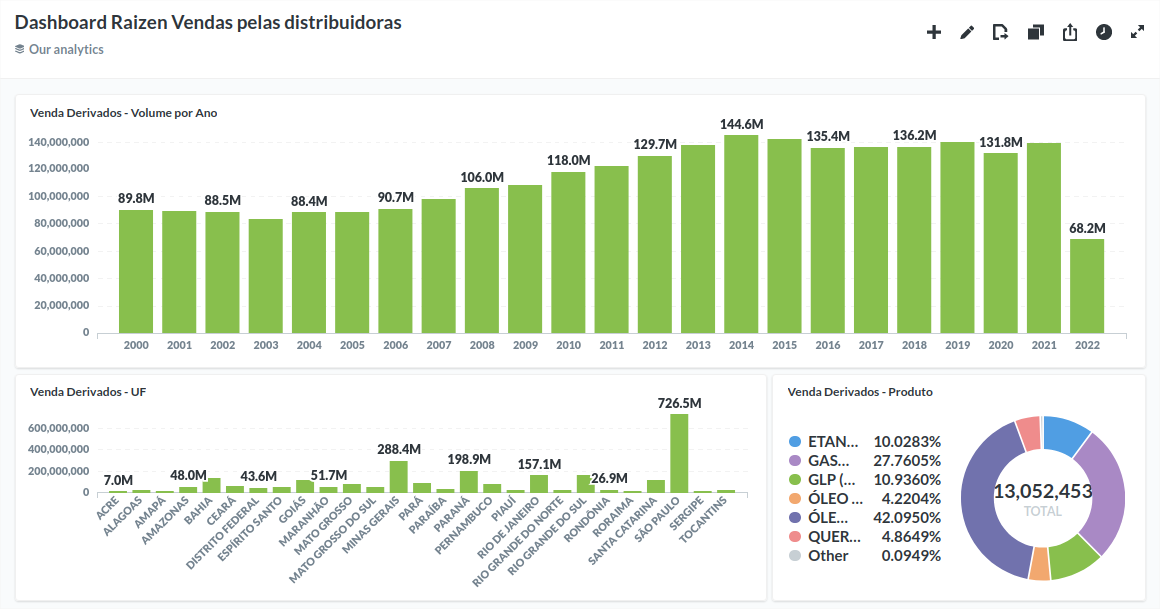
Como vemos a tabela foi criada corretamente, acima dela temos as 2 tabelas do script original sem o uso do airbyte.



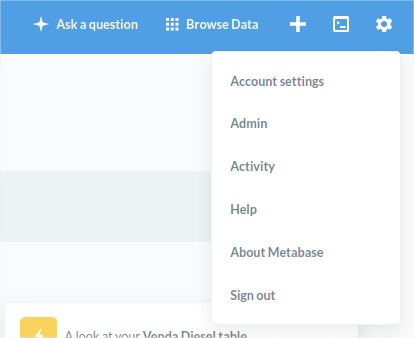
Feito isto devemos fazer o mesmo procedimento novamente mas agora com o arquivo venda\_derivados.csv.

### **Metabase - Configuração e Funcionamento**

O Metabase é uma ferramenta open source especialmente interessante para a parte administrativa de uma empresa que precisa entender mais sobre os dados gerados em sua companhia, sem precisar ser um expert no assunto. A ferramenta de business intelligence é capaz de transformar os dados e suas estruturas em algo palpável através de filtros orgânicos e uma estética confortável para a visualização.



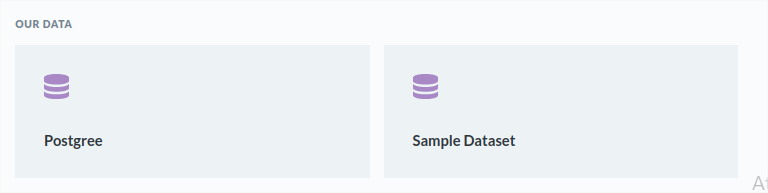
Infelizmente pelo fato do metabase estar no meu localhost não é possível publicar o dashboard. Mas é possível criá-lo de uma maneira muito fácil. Para isso devemos acessar ao metabase através da url -> <http://localhost:3000>

No primeiro acesso será necessário fazer um breve cadastro (pode colocar qualquer dado e prosseguir).B

1. Na home do metabase, precisamos configurar a conexão com o nosso banco de dados postgree. Para isso devemos clicar na engrenagem na parte superior direita e clicar em Admin.
2. Dentro da página de administração devemos clicar em databases e depois clicar em -> **Add database**
3. Dentro da página de configuração devemos escolher o tipo de database no nosso caso -> **PostgreSQL**
4. Escolhemos um nome para a nossa conexão -> **Postgree Metabase**
5. No host colocamos -> 172.17.0.1 (ip interna do docker)
6. Na porta mantemos 5432
7. No database name, username e no password colocamos **docker.**
8. Por último clicamos em save.

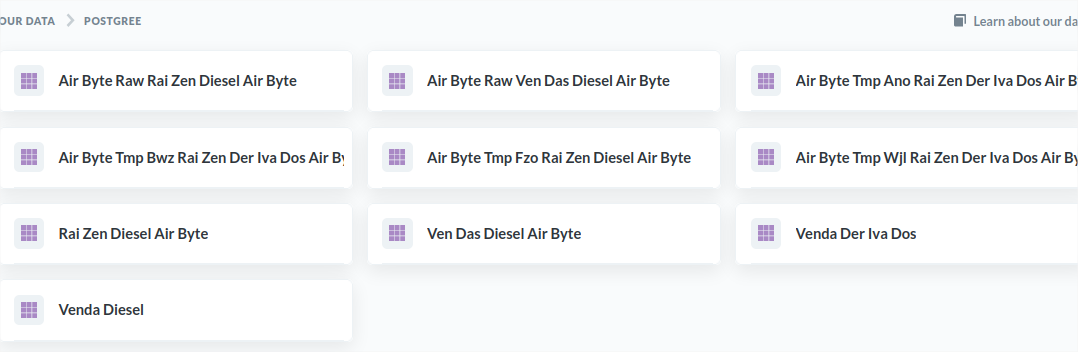
Feito isso voltamos para o <http://localhost:3000>

E já podemos ver o nosso banco de dados com todas as tabelas contidas nele.



#### **Criando um relatório**

Para criar um relatório é bem simples, para isso clicamos no nosso banco de dados que acabamos de importar. Ao clicar nele vai aparecer todas as tabelas contidas dentro do banco. E devemos selecionar a tabela que queremos consumir, neste caso vou optar pela “Venda Diesel”

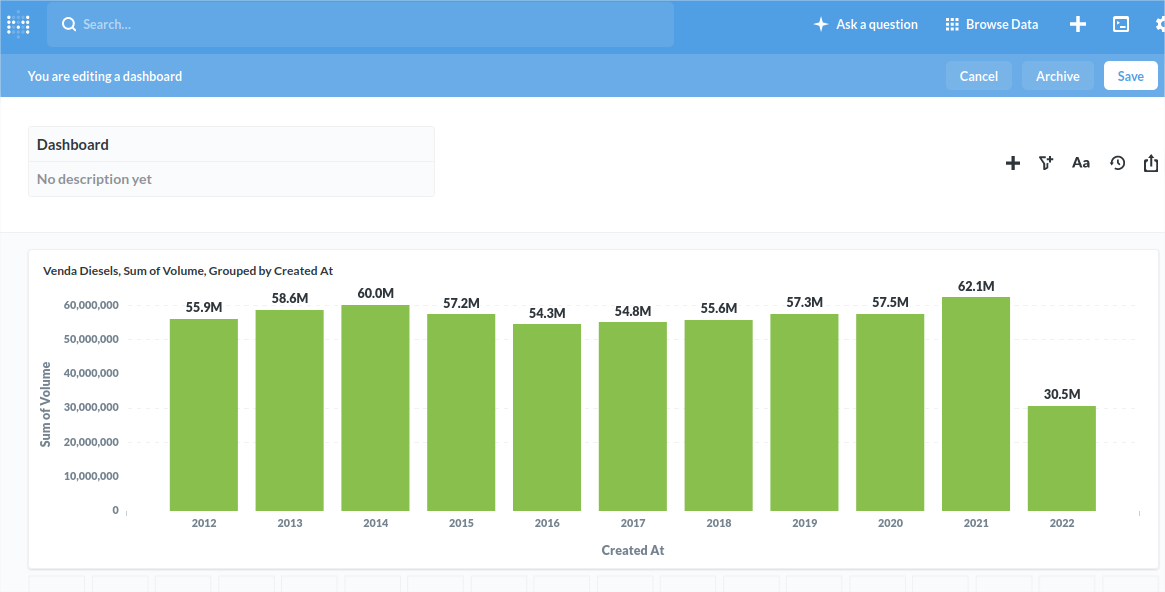


Ao clicar em venda diese uma tabela vai aparecer contendo os dados. Para criar uma visualização simples basta clicar em “Summarize” 

Feito isso uma aba lateral vai abrir e devemos selecionar os dados que queremos mostrar na visualização. Neste caso quero criar uma view que mostra as vendas somadas por ano.

Para isso na seção Summarize by vou alterar o count pela soma do volume. E vou agrupar pela coluna created\_at. Ao fazer isso, a tabela irá mudar para a visualização feita.

Com a exibição pronta, clicamos em Done e depois em Save. Damos um nome para ela e salvamos. Ao fazer isso o metabase irá perguntar se queremos adicionar a um dashboad, clicamos em Sim! Na nova aba clicamos em “Create a new Dashboard”, damos um nome a ele e criamos. Feito isso, a nossa visualização será criada dentro do nosso dashboard.



Para criar outros dashboards basta seguir a mesma lógica com outras colunas ou tabelas.

### **Usuários e senhas**

* Postgree
  + Usuário: docker
  + Senha: docker
  + Db: docker
* Metabase
  + Usuário criado ao subir o docker-compose
* Airflow:
  + Usuário: airflow
  + Senha: airflow