Documentação do desafio técnico - Mevo Farma

Esse relatório tem por objetivo descrever o script criado para o desafio solicitado, utilizando a biblioteca Pandas em Python. Os dados utilizados são de produtos de vários fornecedores que sao carregados a partir de 4 arquivos .CSV, complementando essas informações com dados adicionais de uma API e, por fim, consolida todas as informações em um único arquivo CSV. Abaixo seguem as etapas da criação do código:

1. Importações de Bibliotecas

O código começa importando duas bibliotecas: **Pandas,** para realizar a análise de dados, e **Requests**, para realizar solicitações HTTP para resposta da API.

```
import pandas as pd
import requests
```

2. Definição da Função `get_additional_info`

A função **get_additional_info** recebe um **product_id** um **token**. Ela faz uma solicitação **GET** para a **API**, passando o **product_id** e o **token** em autorização no cabeçalho. Após a execução com sucesso, retorna informações como peso, preço e imposto do produto. Se ocorrer algum erro na solicitação, retorna **None**.

```
def get_additional_info(product_id, token):
    try:
        response =
requests.get(f'https://api.produtos.com/informacoes/{product_id}',
headers={'Authorization': f'Bearer {token}'})
        response.raise_for_status()
        data = response.json()
        return data.get('peso', None), data.get('preco', None),
data.get('imposto'  None)
    except requests.exceptions.RequestException as e:
        return None  None  None
```

3. Definição da Função `load_data_from_csv` e lista de arquivos .CSV

Essa função recebe uma lista de nomes de arquivos CSV como entrada. Ela lê cada arquivo CSV usando o **pd.read_csv()** e, em seguida, concatena todos os **DataFrames** em um único **DataFrame** usando **pd.concat()**. O DataFrame final contém todos os dados dos arquivos CSV fornecidos. Além disso, é informado também o link para WS para acesso a API.

```
def load_data_from_csv(arquivos):
    return pd.concat([pd.read_csv(arquivo) for arquivo in arquivos])
```

```
arquivos = ['fornecedor1.csv'  'fornecedor2.csv'  'fornecedor3.csv'
'fornecedor4.csv']
```

```
token = 'token_chamada_api'
```

4. Carregamento e Remoção de Duplicatas dos Dados CSV

Os dados dos arquivos CSV importados no código são carregados e armazenados em um DataFrame **df.** Em seguida, são removidas as linhas duplicadas usando **drop_duplicates(),** para garantir a integridade dos arquivos.

```
df = load_data_from_csv(arquivos).drop_duplicates()
```

5. Obtendo Informações Sugeridas da API

Para cada **product_id** no DataFrame **df**, as informações são obtidas da API usando a função **get_additional_info()**. Os resultados são armazenados em uma lista **additional_info** e, em seguida, são convertidos em um DataFrame **additional_df**.

```
additional_info = [get_additional_info(product_id, token) for
product_id in df['ID']]
additional_df = pd.DataFrame(additional_info, columns=['peso',
'preco', 'imposto'])
```

6. Concatenação dos DataFrames e Gravação dos dados consolidados em arquivo CSV

Os DataFrames são concatenados em tabela utilizado **pd.concat()**. O DataFrame resultante é armazenado em **final_df**. Após o DataFrame consolidado é salvo em um novo arquivo CSV chamado **'produtos_final.csv'** usando o método **to_csv().**

```
final_df = pd.concat([df, additional_df], axis=1)
```

```
final_df.to_csv('produtos_final.csv', index=False)
```

7. Mensagem de conclusão (Opcional)

O código abaixo indica a conclusão de um processo, ou seja, é exibida uma mensagem indicando que o arquivo 'produtos_final.csv' foi criado com sucesso.

```
print("Processo concluído. Arquivo 'produtos_final.csv' criado com sucesso.")
```