Processo Seletivo EDJR - Otávio Soares Mortosa

Problema 1:

A captura dos dados esperados foi feita através de um algoritmo escrito na linguagem Python. Foram utilizadas as bibliotecas: selenium, bs4, request, pandas e time. As seguintes funções foram designadas ao uso de cada uma:

- selenium - permite acessar a página através de um browser específico. Utilizou-se o Google Chorme;
- bs4 - em específico, utilizou-se o método BeautifulSoup, o qual permite o processamento do HTML da página, para retirar os campos onde estão as informações desejadas;
- request - retorna o HTML de uma página, agregando em praticidade na etapa utilizada;
- pandas - possibilita a manipulação de dados de forma efetiva, criando Data Frames e permitindo a conversão para arquivos de várias extensões (como a desejada);
- time - permite medir o tempo. Utilizada para medir o tempo de execução do programa.

O código está dividido nas seguintes funções:

- coleta_links(url_entrada) - retorna os links das páginas resultantes da pesquisa feita de acordo com o período desejado. Tem como argumento o link inicial da pesquisa;
- adiciona_instancia(resolucao) - adiciona uma instância na variável global “dados”, para registrar os dados coletados em cada iteração daquela etapa. Tem como argumento o campo “resolucao”, uma vez que uma mesma resolução possuirá vários registros em “dados”;
- coletor_dados(publicacoes) - a função que de fato coleta os dados. Faz a varredura no HTML de cada página e trata os dados de interesse a serem coletados. Tem como argumento “publicacoes”, o retorno da função coleta_links().
- gera_excel - função simples, utilizando o pandas, para gerar o arquivo de extensão .xlsx com os dados coletados, na formatação desejada.

Obs.: mais detalhes técnicos sobre o código se encontram nos comentários do código.

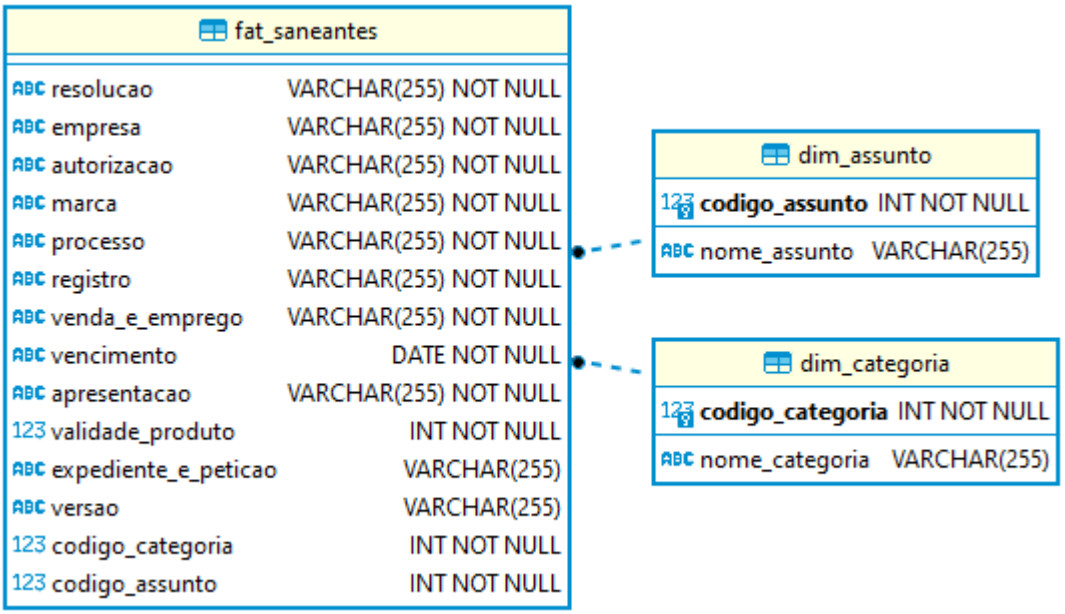
Como resultado, foram encontradas 6 publicações (links) na pesquisa, as quais, no total, geraram 243 entradas no arquivo final .xlsx. A execução total do código foi de 1.125 segundos, como apresentado abaixo:

```
PS C:\Users\otavi\Processo-Seletivo-EDJR>  c:; cd 'c:\Users\otavi\Processo-Seletivo-EDJR'; & 'C:\Us
\python\debugpy\adapter/../..\debugpy\launcher' '57750' '--' 'c:\Users\otavi\Processo-Seletivo-EDJR'

DevTools listening on ws://127.0.0.1:57756/devtools/browser/cf2e68e3-c115-4554-9aa1-8a5fc36063dc
--- 1.125 seconds ---
PS C:\Users\otavi\Processo-Seletivo-EDJR>
```

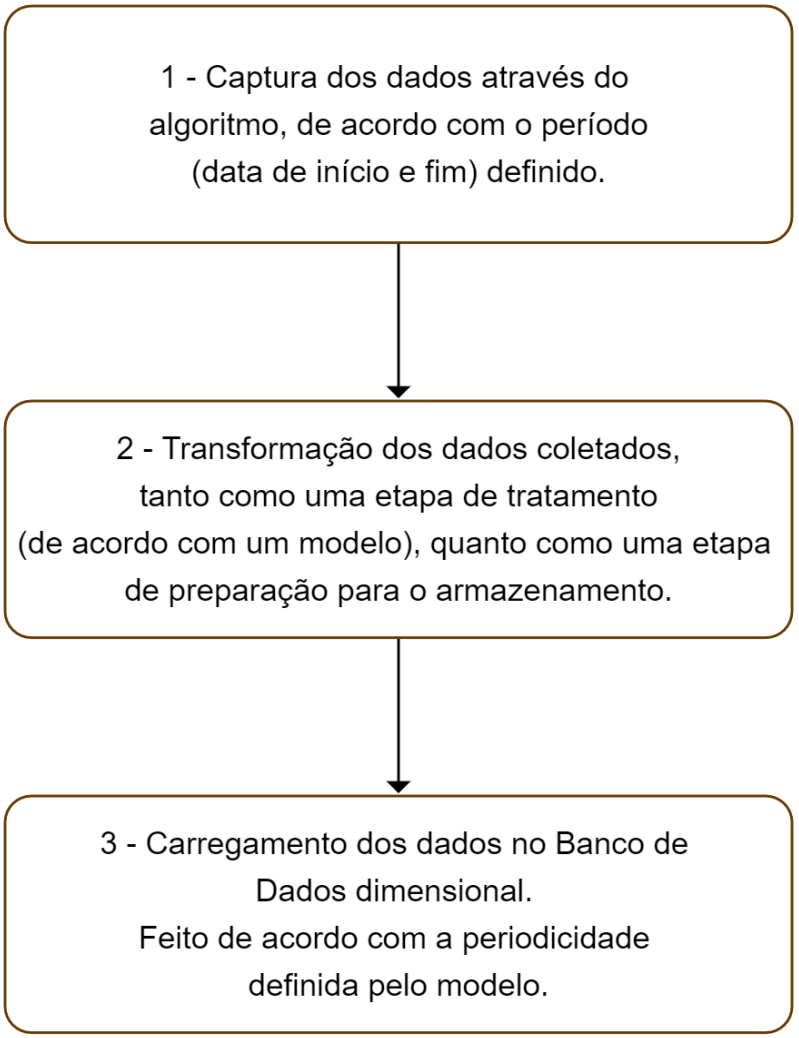
Problema 2:

O seguinte modelo (dimensional) dos dados foi feito:



Obs.: um arquivo pdf do mesmo modelo, mas com apresentação visual diferente, de nome “modelo dimensional saneantes”, foi anexado à pasta enviada.
Como sugestão, ainda é possível criar tabelas de dimensão para outros campos, os quais dizem a respeito de uma transação, como “empresa”, “data” e “venda_e_emprego”.

Para a segunda parte do problema, o seguinte esquema processual é proposto:



Para o passo número 1, utilizaria-se um código de extração (similar ao produzido) para a coleta dos dados. Observa-se que a url inicial é necessária para cada execução, uma vez que ela corresponde ao período de interesse da coleta de dados. Para o passo número 2, uma forma de organização similar, ou mais elaborada, implementada no código apresentado, poderia ser feita. O que se deve ressaltar é que o modelo em que se espera os dados é determinante (formato, tratamento ou arquivo de saída). Como a etapa anterior, essa (2) está diretamente ligada à próxima, uma vez que sua saída será o que irá popular o Banco de Dados modelado e implementado. Finalmente, para o passo número 3, é modelado um Banco de Dados dimensional de acordo com as necessidades do modelo de negócio e forma dos dados coletados. Esta etapa poderia ser implementada como uma continuação do código apresentado, uma vez que a linguagem Python possui bibliotecas de manipulação de banco de dados, juntamente com ferramentas específicas de gerenciamento de Bancos de Dados e visualização, como o Qlik.