Relatório: Desafio Pipeline de Dados - Secretaria de Saúde do Recife

Vaga Engenheiro de Dados Júnior - GGMA

Data: 31/01/2025

Candidato: João Paulo Oliveira Nolasco

Índice

---

1. Introdução

Este relatório descreve o processo de construção de um pipeline de dados para processamento e análise de informações sobre medicamentos disponibilizados pela Prefeitura do Recife. O objetivo é extrair dados de um portal público, transformá-los para adequação analítica e carregá-los em um banco de dados para facilitar consultas e análises posteriores. O pipeline segue a arquitetura Medalhão, que organiza os dados em camadas progressivamente mais refinadas: Bronze, Silver e Gold.

---

2. Descrição dos Dados

2.1 Fontes de Dados

Os dados utilizados foram extraídos do portal de dados públicos da Prefeitura do Recife, especificamente do conjunto de dados sobre **medicamentos distribuídos no município**. O arquivo contém informações como nome do produto, quantidade distribuída, bairro de distribuição, classe do medicamento, entre outras.

Link para a fonte dos dados no site: Portal de Dados Recife - Medicamentos

Link do CSV:
http://dados.recife.pe.gov.br/dataset/443797b4-5c9c-421c-8509-62eb7e6d2fc9/resource/4c52c602-6b6f-4ca5-bcb9-a64248578058/download/medicamentos.csv

2.2 Motivo da Escolha dos Dados

Escolhi esse conjunto de dados por sua relevância pública, pois ele fornece informações cruciais sobre a distribuição de medicamentos pela Secretaria de Saúde do Recife. Este dataset não só aborda um tema de importância social, como também oferece dados que fazem sentido para um exercício prático de engenharia de dados, como simulado em um ambiente real de trabalho numa secretaria de saúde.

---

3. Arquitetura do Pipeline de Dados

3.1 Arquitetura Medalhão

A arquitetura Medalhão é uma metodologia de organização de dados que divide o processo de transformação em três camadas distintas: Bronze, Silver e Gold. Cada camada tem um papel específico:

- **Bronze**: Armazena os dados brutos, sem transformações. Esta camada serve como o ponto inicial de armazenamento após a extração.
- **Silver**: Os dados nesta camada são transformados, limpos e preparados, mas não estão totalmente prontos para análise final.
- **Gold**: Dados analíticos e altamente processados, prontos para consultas e visualizações.

3.2 Como a Arquitetura Medalhão foi Aplicada

A arquitetura Medalhão foi aplicada da seguinte forma:

- **Camada Bronze**: Dados foram extraídos diretamente do arquivo CSV e armazenados em uma tabela chamada bronze_table no banco PostgreSQL.
- **Camada Silver**: Após o carregamento na camada Bronze, os dados passaram por transformações como limpeza e normalização, e foram armazenados no schema silver.
- **Camada Gold**: Consultas analíticas foram realizadas para gerar informações mais detalhadas, como a distribuição de medicamentos por bairro e classe, e os resultados foram armazenados no schema gold.

---

4. Ambiente e Ferramentas Utilizadas

Para a realização deste projeto, diversas ferramentas e tecnologias foram empregadas. A seguir, descrevo o ambiente utilizado para desenvolvimento e execução do pipeline de dados:

Docker

Utilizei o **Docker** para garantir que o ambiente de desenvolvimento fosse isolado, reproduzível e facilmente escalável. O banco de dados **PostgreSQL** foi configurado e executado dentro de um container Docker. Isso permitiu uma instalação limpa e a possibilidade de replicar o ambiente em qualquer máquina com Docker instalado, sem depender de configurações manuais de banco de dados.

PostgreSQL

O **PostgreSQL** foi utilizado como sistema de gerenciamento de banco de dados relacional (SGBD). Ele foi configurado para armazenar os dados em três camadas distintas: **Bronze**, **Silver** e **Gold**. Cada camada tem um propósito específico no processo de transformação e análise dos dados. O PostgreSQL foi integrado ao pipeline de dados usando a biblioteca **SQLAlchemy**.

SQLAlchemy

A **SQLAlchemy** foi utilizada para facilitar a conexão entre o Python e o banco de dados PostgreSQL. Essa biblioteca fornece uma camada de abstração para realizar consultas SQL e interagir com o banco de dados de forma eficiente e sem a necessidade de escrever SQL diretamente em todas as partes do código.

Bibliotecas Python

As bibliotecas Python utilizadas no projeto foram listadas no arquivo **requirements.txt**, o qual facilita a instalação de todas as dependências necessárias para rodar o pipeline. As principais bibliotecas utilizadas foram:

- **Pandas**: Para manipulação e transformação dos dados.
- **SQLAlchemy**: Para a interação com o banco de dados PostgreSQL.
- **Requests**: Para realizar a requisição HTTP e obter o arquivo CSV.
- **IPython**: Para exibição de DataFrames durante o processo de transformação.

Ambiente de Desenvolvimento

O projeto foi desenvolvido em um ambiente local, utilizando o **Jupyter Notebook** para escrever o código e testar as transformações de dados. A execução do pipeline e as consultas SQL foram realizadas diretamente através do notebook, o que permitiu maior interatividade durante o desenvolvimento.

5. Processo de Extração e Transformação

5.1 Extração dos Dados

A extração foi realizada a partir de um arquivo CSV hospedado no portal de dados da Prefeitura. Utilizando a biblioteca requests, fiz uma requisição HTTP para obter o arquivo CSV. Os dados foram lidos usando pandas e carregados diretamente na camada Bronze do banco PostgreSQL.

5.2 Transformação dos Dados

Na camada **Silver**, diversas transformações foram realizadas para garantir que os dados estivessem prontos para análises mais profundas. As transformações feitas incluem:

- **Limpeza de Caracteres Especiais**: Algumas colunas de texto continham caracteres especiais indesejados, o que poderia afetar a consistência dos dados. Para tratar esses casos, utilizei a função fix_special_characters que garantiu que os dados estivessem no formato correto, sem caracteres que poderiam gerar erros ou dificultar análises posteriores.

- **Substituição de Valores Nulos**: Durante a transformação, valores nulos (NaN) foram tratados adequadamente. Para colunas de texto, substituí valores nulos por “Sem informação”, assegurando que não houvesse lacunas que pudessem prejudicar a interpretação dos dados. Já para colunas numéricas, valores ausentes foram preenchidos com zero, garantindo a integridade dos cálculos posteriores.

- **Conversão de Tipos de Dados**: Algumas colunas exigiam conversão de tipos de dados para facilitar análises e garantir que os dados estivessem corretamente tipados:
    - A coluna **quantidade**, originalmente em formato de texto, foi convertida para **int** após a limpeza de vírgulas.

    - Colunas que representavam códigos ou identificadores, como **distrito**, **cadum**, e **codigo_produto**, foram convertidas para o tipo **str** para manter sua consistência.
    - A coluna **data_carga** foi convertida para o tipo **datetime** para garantir que todas as datas estivessem corretamente formatadas e facilitassem qualquer análise temporal futura.

- **Normalização dos Nomes das Colunas**: Para evitar erros e melhorar a legibilidade do código, todos os nomes das colunas foram normalizados, removendo espaços e colocando-os em minúsculas. Isso facilitou o trabalho ao referenciar as colunas no código, além de reduzir as chances de erros de digitação.

- **Criação de ID (Chave Primária)**: Uma melhoria importante foi a criação da coluna **ID** para garantir a unicidade dos registros. Cada linha do dataset recebeu um identificador único, o que facilita a referência dos dados em operações futuras, como joins e consultas, além de permitir maior controle sobre os registros.

- **Criação das Colunas de Timestamps (DT_CREATED e DT_UPDATED)**: As colunas de timestamp foram criadas em todas as camadas (Bronze, Silver e Gold), com valores de data e hora registrados no momento da carga ou atualização dos dados. Essas colunas são essenciais para monitorar quando os dados foram carregados ou alterados, garantindo a rastreabilidade do processo de ETL e facilitando a manutenção do pipeline no futuro.

Após a realização dessas transformações, os dados foram armazenados na camada **Silver**, prontos para análises mais detalhadas, e seguiu para a camada **Gold**, onde seriam agregados e analisados.

---

6. Consultas

6.1 Descrição das Consultas SQL

Duas consultas principais foram realizadas na camada Silver e os resultados foram carregados no schema Gold:

- **Consulta por Bairro**: Agrupa os dados por bairro, somando a quantidade de medicamentos distribuídos e contando o número de produtos distintos por bairro.

```
Bairros com dados detalhados salvos no schema gold.
```

	bairro	total_quantidade	total_produtos_distintos	total_classes_presentes	dt_created	dt_updated
0	AFOGADOS	3479687.0	606	119	2025-01-31 13:16:19.696926+00:00	2025-01-31 13:16:19.696926+00:00
1	IPSEP	2784518.0	503	91	2025-01-31 13:16:19.696926+00:00	2025-01-31 13:16:19.696926+00:00
2	IBURA	1199774.0	518	120	2025-01-31 13:16:19.696926+00:00	2025-01-31 13:16:19.696926+00:00
3	MADALENA	1162785.0	450	97	2025-01-31 13:16:19.696926+00:00	2025-01-31 13:16:19.696926+00:00
4	COHAB	1137067.0	254	76	2025-01-31 13:16:19.696926+00:00	2025-01-31 13:16:19.696926+00:00
...	...	...	...	...	...	...
66	ALTO DO MANDU	18319.0	245	83	2025-01-31 13:16:19.696926+00:00	2025-01-31 13:16:19.696926+00:00
67	AF0GADOS	17872.0	101	41	2025-01-31 13:16:19.696926+00:00	2025-01-31 13:16:19.696926+00:00
68	ROSARINHO	16176.0	116	44	2025-01-31 13:16:19.696926+00:00	2025-01-31 13:16:19.696926+00:00
69	ENCRUZILHADA	10899.0	385	92	2025-01-31 13:16:19.696926+00:00	2025-01-31 13:16:19.696926+00:00
70	HIPODROMO	22.0	53	28	2025-01-31 13:16:19.696926+00:00	2025-01-31 13:16:19.696926+00:00

71 rows × 6 columns

- **Consulta por Classe de Produto**: Agrupa os dados por classe de produto, mostrando o número de produtos disponíveis e o número de bairros atendidos por cada classe.

```
Produtos por classe com dados detalhados salvos no schema gold.
```

	classe	total_produtos_distintos	total_bairros_atendidos	total_registros_classe	dt_created	dt_updated
0	Sem informação	732	71	14268	2025-01-31 13:16:19.904340+00:00	2025-01-31 13:16:19.904340+00:00
1	ANTIEPILÉPTICO	10	39	448	2025-01-31 13:16:19.904340+00:00	2025-01-31 13:16:19.904340+00:00
2	REPOSITOR HIROELETROLÍTICO	10	71	703	2025-01-31 13:16:19.904340+00:00	2025-01-31 13:16:19.904340+00:00
3	ANALGÉSICO OPIÓIDE	9	39	208	2025-01-31 13:16:19.904340+00:00	2025-01-31 13:16:19.904340+00:00
4	ANTI-INFLAMATÓRIO (ESTERÓIDE)	8	71	1101	2025-01-31 13:16:19.904340+00:00	2025-01-31 13:16:19.904340+00:00
...	...	...	...	...	...	...
118	ANESTÉSICO HIPNÓTICO	1	11	12	2025-01-31 13:16:19.904340+00:00	2025-01-31 13:16:19.904340+00:00
119	ANTITIREOIDIANO	1	37	55	2025-01-31 13:16:19.904340+00:00	2025-01-31 13:16:19.904340+00:00
120	ANTIVERTIGINOSO	1	65	153	2025-01-31 13:16:19.904340+00:00	2025-01-31 13:16:19.904340+00:00
121	CALMANTE (FITOTERÁPICO)	1	63	134	2025-01-31 13:16:19.904340+00:00	2025-01-31 13:16:19.904340+00:00
122	COAGULANTE/ HEMOSTÁTICO	1	7	8	2025-01-31 13:16:19.904340+00:00	2025-01-31 13:16:19.904340+00:00

123 rows × 6 columns

6.2 Justificativa para Escolha das Consultas

As consultas foram escolhidas para fornecer um recorte interessante e pertinente para um posterior trabalho de analistas/cientistas de dados. A ideia aqui foi realmente deixar uma consulta bem aberta e com possibilidades de análises sobre elas, e não já fazer a query com a análise minuciosa já feita. As querys trazem recortes sobre a distribuição geográfica dos medicamentos e a diversidade de classes de produtos. A consulta por bairro permite identificar áreas com maior demanda de medicamentos,

enquanto a consulta por classe ajuda a entender quais tipos de medicamentos são mais distribuídos. Essas consultas oferecem uma visão clara e detalhada do panorama da distribuição de medicamentos na cidade.

7. Conclusão

Este projeto de construção de um pipeline de dados foi uma excelente oportunidade para aplicar conceitos fundamentais de engenharia de dados em um contexto real, usando dados sobre medicamentos distribuídos pela Prefeitura do Recife.

A **arquitetura Medalhão** foi adotada para estruturar o pipeline em três camadas distintas: Bronze, Silver e Gold. Isso permitiu organizar os dados de forma eficiente, com a camada Bronze armazenando os dados brutos, a Silver contendo dados tratados e a Gold sendo a camada final para consultas analíticas. Esse processo de organização proporcionou uma estrutura robusta e escalável, ideal para análise de dados.

As transformações realizadas nos dados incluíram a limpeza de valores nulos, normalização de colunas e ajustes de tipos de dados. Essas transformações foram fundamentais para garantir que os dados estivessem prontos para análise detalhada.

---

8. Reflexões Finais

Um desafio importante foi a conciliação entre o desenvolvimento do projeto e as demandas acadêmicas e profissionais, visto que estava simultaneamente envolvido em atividades acadêmicas e trabalho. No lado técnico, a maior dificuldade foi garantir que o projeto atendesse aos requisitos especificados, além do uso de ferramentas e bibliotecas que não tinha costume de utilizar.

Além disso, o uso do Docker para rodar o banco de dados PostgreSQL foi uma experiência interessante, mas que trouxe desafios na configuração do ambiente. A utilização de containers facilitou o isolamento do ambiente de desenvolvimento, mas a configuração inicial exigiu um esforço adicional. Por fim, um desafio enfrentado que é muito comum quando se trabalha com projetos é a definição do escopo, então preferi fazer algo mais simples porém que tivesse domínio, do que implementar e testar ideias e ferramentas que possivelmente não teria tempo hábil de executar corretamente no projeto.

Uma melhoria importante para o projeto que eu queria implementar caso tivesse mais tempo seria a **automação do processo de atualização dos dados**. Isso garantiria que o pipeline fosse sempre atualizado com as informações mais recentes, sem a necessidade de intervenção manual. A automação poderia ser realizada por meio de **orquestração de pipelines de dados**, utilizando ferramentas como **Apache Airflow** ou **Prefect**. Essas ferramentas permitem agendar, monitorar e gerenciar a execução dos processos ETL de maneira eficiente, além de garantir que novas atualizações dos dados sejam feitas automaticamente, conforme o arquivo original no portal da Prefeitura for atualizado.

A **orquestração** também ajudaria a coordenar as diversas etapas do pipeline (extração, transformação e carregamento), proporcionando uma solução mais robusta, confiável e escalável. Dessa forma, o pipeline se tornaria mais autossuficiente, com maior eficiência e menos necessidade de supervisão constante.