**Documentação do Projeto - Avaliação Técnica de Analista de Dados**

**Introdução**

Este documento descreve a lógica e a arquitetura utilizadas na construção do dashboard de análise de vendas, desenvolvido como solução para a avaliação técnica. O projeto foi estruturado em três camadas principais: Backend (armazenamento), Processamento (análise) e Frontend (apresentação).

**Primeira Camada: Backend - PostgreSQL**

A primeira camada do projeto, o backend, foi construída em PostgreSQL com foco na **estruturação e integridade dos dados**. A lógica principal consistiu em armazenar os dados brutos de forma organizada e eficiente, seguindo as boas práticas de normalização de tabelas para garantir a confiabilidade e evitar a redundância de informações.

Para a criação dos dados mockados, busquei inspiração em modelos de varejo de grande porte, como Americanas e Havan, que possuem uma vasta diversidade de categorias de produtos. Essa abordagem permitiu simular um cenário de vendas mais realista e abrangente.

Foram criadas quatro tabelas distintas (clientes, produtos, pedidos, itens\_pedido), estabelecendo os relacionamentos entre elas conforme solicitado no teste, para formar um modelo de dados coeso. Uma anomalia de dados foi intencionalmente inserida para atender ao requisito da **Questão 7**. Em resumo, o objetivo desta camada foi construir uma base de dados sólida e confiável como alicerce para todas as análises subsequentes.

**Segunda Camada: Processamento - Python, SQL e Pandas**

A segunda camada é responsável pelo **processamento e transformação dos dados brutos em insights**. Ela atua como o cérebro da aplicação, conectando o frontend ao backend.

* **Conexão e Execução (Python/psycopg2):** A função run\_query foi criada para centralizar e gerenciar a comunicação com o banco de dados. Sua lógica é receber uma consulta SQL, executá-la de forma segura no PostgreSQL e retornar os resultados estruturados como um DataFrame do Pandas, pronto para a manipulação.
* **Análise (SQL):** A lógica de análise foi expressa através de "perguntas" em SQL, formuladas para resolver cada questão da avaliação. Por exemplo, na **Questão 4** ("Top 5 produtos mais rentáveis"), a estratégia foi:
  1. Criar uma CTE para calcular a receita total de cada produto no último ano.
  2. Na consulta final, unir o resultado com a tabela de produtos para obter os nomes, ordenar pela receita de forma decrescente e limitar o resultado aos 5 primeiros.
* **Manipulação (Pandas):** Com os dados retornados pelo SQL, o Pandas foi utilizado para a lógica de **filtragem e formatação**. No dashboard interativo, por exemplo, o Pandas filtra dinamicamente o DataFrame completo com base nas seleções do usuário na sidebar (datas, categorias, clientes). Além disso, ele foi essencial para formatar dados para a visualização, como a conversão de valores numéricos para o formato de moeda (ex: 70.5 para R$ 70,50).

**Terceira Camada: Frontend - Streamlit**

A camada final é o frontend, desenvolvido com Streamlit para **apresentar os resultados de forma clara e interativa**. Embora este projeto tenha sido uma excelente oportunidade para aprofundar meus conhecimentos na ferramenta, a lógica de desenvolvimento focou em criar uma experiência de usuário intuitiva.

Utilizando um conceito popularizado pelo físico Richard Feynman, busquei explicar os achados de forma simples e direta. A lógica geral do projeto pode ser resumida em um fluxo contínuo:

O **PostgreSQL** armazena os fatos brutos → O **Python/SQL** faz perguntas inteligentes para transformar esses fatos em respostas (insights) → O **Pandas** organiza e filtra essas respostas → O **Streamlit** exibe os insights de forma interativa e permite que o usuário refaça o ciclo através dos filtros.

**Conclusão e Próximos Passos**

As três camadas – estruturar, analisar e apresentar – formam os pilares deste projeto. O resultado final é um dashboard funcional e completo que atende a todos os requisitos propostos na avaliação.

Refletindo sobre o projeto, identifico pontos de evolução que seriam considerados em um ambiente de produção para garantir maior eficiência e escalabilidade:

* **Performance dos Filtros:** Em vez de filtrar os dados no Pandas, eu passaria os critérios de filtro (data, cliente, etc.) diretamente para a consulta SQL, utilizando a cláusula WHERE. Isso faria com que o banco de dados retornasse um volume de dados muito menor, otimizando a performance da aplicação.
* **Gráficos Mais Interativos:** Exploraria bibliotecas como o Plotly para criar visualizações onde o usuário pudesse interagir diretamente com os gráficos (ex: ver valores ao passar o mouse, dar zoom).
* **Estrutura Escalável:** Para um projeto maior, o código seria modularizado em vários arquivos (ex: queries.py, charts.py, app.py) para facilitar a manutenção e o desenvolvimento colaborativo.
* **Expansão da Análise:** Adicionaria novas fontes de dados para análises mais complexas, como condições de pagamento, status de entrega ou dados de crédito para análise de risco.