**Desafio Técnico Databricks: Pipeline de Dados Olist com Notebooks e Workflows**

**Objetivo:**

Construir um pipeline de dados ELT funcional utilizando exclusivamente recursos da plataforma Databricks. O pipeline deverá ingerir dados do dataset Olist, transformá-los através de notebooks Spark (SQL ou PySpark) para criar camadas Bronze e Silver em Delta Lake, e orquestrar a execução ponta-a-ponta com Databricks Workflows (Jobs). O foco é demonstrar a habilidade em construir, validar e orquestrar processos de dados na plataforma Databricks.

**Contexto:**

Utilizaremos um subconjunto de dados do **"Brazilian E-Commerce Public Dataset by Olist"** (Kaggle: https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce).

**Dataset:**

- **Arquivos Essenciais:**
  - `olist_orders_dataset.csv`
  - `olist_order_items_dataset.csv`
  - `olist_customers_dataset.csv`
- **Suposição:** Faça com que os 3 arquivos CSV estejam disponíveis em um local acessível pelo Databricks (ex: DBFS `/mnt/raw_data/olist/` ou um volume do Unity Catalog).

**Ferramentas e Recursos Databricks a serem Utilizados:**

1. **Databricks Notebooks:** Para escrever a lógica de ingestão e transformação usando Spark (SQL ou PySpark).
2. **Apache Spark (via Databricks):** O motor de processamento para as transformações.
3. **Delta Lake:** Formato de armazenamento para as camadas Bronze e Silver.
4. **Databricks Workflows (Jobs):** Para orquestrar a execução sequencial dos notebooks.
5. **(Opcional) Unity Catalog:** Para governança (se familiarizado), caso contrário, usar o metastore padrão (Hive).

**Tarefas (Abordagem com Notebooks Spark):**

1. **Notebook de Ingestão (Bronze Layer):**
   - Crie um Databricks notebook (Python/PySpark recomendado para flexibilidade, SQL aceitável).
   - Leia os 3 arquivos CSV (`orders`, `order_items`, `customers`).
   - Defina ou infira os schemas apropriados; realize conversões básicas de tipos de dados.
   - Escreva os DataFrames resultantes como tabelas Delta no schema "Bronze" (ex: `bronze_olist_orders`, `bronze_olist_order_items`,

`bronze_olist_customers`). Utilize modo `overwrite` para simplificar o processo em lote do desafio.
    - **Entregável:** Código do notebook de ingestão.
2. **Notebook de Transformação de Dimensão (Silver Layer):**
    - Crie um **novo** Databricks notebook.
    - Leia a tabela Delta `bronze_olist_customers`.
    - Aplique transformações: selecione e renomeie colunas, trate dados (ex: limpeza básica), garanta a unicidade do `customer_id`.
    - **Implemente Verificações de Qualidade:** Adicione código Spark no notebook para validar dados essenciais. O notebook deve falhar se a qualidade não for atendida. Nenhum identificador nulo, por exemplo
    - Escreva o resultado na tabela Delta `silver_dim_customers`.
    - **Entregável:** Código do notebook de transformação da dimensão.
3. **Notebook de Transformação de Fatos (Silver Layer):**
    - Crie um **novo** Databricks notebook.
    - Leia as tabelas Delta `bronze_olist_order_items`, `bronze_olist_orders` e `silver_dim_customers`.
    - Realize os joins necessários para combinar as informações.
    - Selecione as colunas relevantes para a tabela `fct_order_items`.
    - Calcule métricas básicas, se necessário (ex: `preco_total_item = price + freight_value`).
    - **Implemente Verificações de Qualidade:** Adicione código Spark para validar dados. Falhe o notebook se as verificações não passarem.
    - Escreva o resultado na tabela Delta `silver_fct_order_items`.
    - **Entregável:** Código do notebook de transformação de fatos.
4. **Orquestração com Databricks Workflows (Jobs):**
    - Crie um Databricks Job.
    - Adicione tarefas ao Job para executar os notebooks das Tarefas 1, 2 e 3 **na sequência correta**.
    - **Configure Dependências:** Garanta que o notebook da Tarefa 2 só execute após o sucesso da Tarefa 1, e o da Tarefa 3 só após o sucesso da Tarefa 2 (Task Dependencies).
    - Defina um gatilho simples para o Job (ex: manual ou diário).
    - **Entregável:** Captura de tela ou descrição da configuração do Job, mostrando as tarefas e suas dependências e exportação do yml do workflow
5. **Documentação (README.md):**
    - Crie um arquivo `README.md`.
    - Descreva a arquitetura do pipeline (quais notebooks são executados, quais tabelas são criadas).
    - Explique como configurar (se necessário) e executar o Databricks Job.
    - Detalhe as verificações de qualidade implementadas em cada notebook.
    - Forneça exemplos de queries SQL para consultar e verificar os dados nas tabelas `silver_dim_customers` e `silver_fct_order_items`.

    - **(Opcional):** Discuta brevemente como o pipeline poderia ser adaptado para processamento incremental (ex: usando Structured Streaming, Delta MERGE, etc.).
    - **Entregável:** Arquivo `README.md`.

**Critérios de Avaliação:**

- **Funcionalidade do Pipeline:** O Job executa todos os notebooks na ordem correta? As tabelas Bronze e Silver são criadas e populadas corretamente em Delta Lake?
- **Qualidade do Código Spark (Python/SQL):** O código nos notebooks é claro, eficiente e bem organizado? Segue boas práticas de programação Spark?
- **Implementação de Qualidade de Dados:** As verificações de qualidade foram implementadas de forma eficaz dentro dos notebooks? Elas são capazes de interromper o fluxo em caso de problemas?
- **Orquestração e Dependências:** O Databricks Job está corretamente configurado com as dependências entre tarefas?
- **Uso de Recursos Databricks:** O candidato demonstra bom uso de Notebooks, Spark, Delta Lake e Workflows para construir a solução?
- **Documentação:** O `README.md` é claro, completo e útil?

**O que *NÃO* é o Foco Principal (Devido ao Tempo):**

- Implementação completa de processamento incremental (discussão conceitual é suficiente).
- Otimizações de performance extremas (código funcional e razoavelmente eficiente é o esperado).
- Tratamento exaustivo de todos os cenários de erro ou qualidade de dados (foco nas verificações essenciais).