**Case de Engenharia de Dados – Databricks**

**Contexto (texto do cliente — preservado)**

Sua empresa recebe dados transacionais de diferentes bancos parceiros. Esses dados chegam no Data Lake em arquivos **parquet** com streaming ou estão disponíveis para coleta a partir de conexão com o banco de dados do sistema.  
O objetivo é construir uma arquitetura no Databricks que permita:

1. Ingestão confiável em camadas.
2. Controle e monitoramento das cargas em modo **batch**, com parametrização (ex: refresh\_rate, execution\_date).
3. Otimização de **recursos financeiros**.
4. Catálogo de dados com **lineage** e principais relacionamentos entre as entidades.
5. Apresentar indicadores de **baixa latência** para o usuário final.
6. Temos tabelas com cadastro de clientes que podem ser alteradas todos os dias e temos tabelas com transações cujo status pode também ser alterado com o tempo. Precisamos apresentar uma **visão histórica** dos dados e poder refletir a posição atual de forma otimizada.

**Tarefa do candidato — Enunciado**

**Apresente um ambiente Databricks que atenda as necessidades acima.** Tente aplicar uma solução para todos os pontos. Entregáveis obrigatórios:

1. **Diagrama** do ambiente de dados (use <https://app.diagrams.net/>). Deve mostrar componentes: ingestão (batch & streaming), armazenamento (raw/bronze, trusted/silver, curated/gold), catálogos, processo de orquestração, monitoramento, e pontos de otimização de custo.
2. **Esboço de código em PySpark** que mostre a evolução dos dados vindo de uma fonte **streaming** para a primeira camada (bronze/raw) considerando updates/CDC (ou pelo menos idempotência). O código deve contemplar checkpointing e mecanismos para evitar duplicação.
3. **Documento explicativo** (máx. 2–3 páginas) respondendo explicitamente os itens abaixo:
   * Vantagens da arquitetura proposta e se usaria ou não o **modelo medalhão** (bronze/silver/gold).
   * Como implementar o controle para evitar **duplicação ou ausência de dados**.
   * Quando e por que usar **Delta**, **Iceberg** ou **apenas Parquet** (prós/cons) para garantir governança e evolução de dados.
4. (Opcional, pontuação extra) Um notebook simples com exemplo de **MERGE INTO** demonstrando upsert incremental de uma tabela de clientes (cadastro) e de uma tabela de transações (estado muda ao longo do tempo).