Desafio Técnico para Engenheiro(a) de Dados Sênior

Este desafio busca avaliar suas competências técnicas e a sua aptidão para desenvolver soluções de engenharia de dados robustas e escaláveis para problemas complexos. Será demonstrada a sua proficiência em ferramentas como Databricks, bem como a sua capacidade de conceber e documentar uma arquitetura de dados abrangente.

O Desafio Integrado

Este desafio é composto por duas partes interligadas que culminam em uma entrega única como um projeto no GitHub.

Parte 1: Implementação Prática com Databricks

Nesta parte, você demonstrará suas habilidades práticas de engenharia de dados utilizando o Databricks Community Edition.

Os arquivos necessários e as instruções da Parte 1 também se encontram em https://github.com/winnin/desafio_dataeng

Instruções para a Implementação Prática

1. **Configuração do Databricks:**
    - Use o Databricks Community Edition. Se necessário, cadastre-se em https://www.databricks.com/try-databricks#account e escolha "Get started with Community Edition".
    - Ao fazer login, crie um cluster no menu "Compute".
    - Para carregar os arquivos e criar tabelas a partir deles, use o menu "Catalog" -> "Create Table" -> "Drop Files To Upload, or click to browse". Após o upload, o path do arquivo será exibido.
2. **Arquivos para Carregar:**
    - `1-wiki_pages.json.gz`: Contém nomes de criadores de conteúdo (wiki_page) para buscar o user_id do YouTube na API da Wikipedia.
    - `2-posts_creator_json.json.gz`: Contém dados de posts de criadores de conteúdo (creator_id, views, likes, title, published_at, tags, yt_user).
3. **Exercícios:**
    - Crie o notebook "1 - create_table_creators_scrape_wiki" para ler `1-wiki_pages.json.gz` e criar a tabela delta `default.creators_scrape_wiki`.
    - Crie o notebook "2 - create_table_posts_creator" para ler `2-posts_creator_json.json.gz` e criar a tabela delta `default.posts_creator`.
    - Crie o notebook "3 - create_table_user_yt_from_wikipedia_api" para gerar a

tabela delta `default.users_yt`.

- Utilize a tabela `default.creators_scrape_wiki` para buscar o `user_id` do YouTube na API da Wikipedia para cada `wiki_name`.
- Dica 1: Utilize o endpoint https://en.wikipedia.org/w/api.php.
- Dica 2: Utilize os parâmetros `params = {"action": "parse", "page": f"{page_name}", "format": "json"}`.
- Os campos da tabela `default.users_yt` devem ser `user_id` (extraído da Wikipedia) e `wiki_page` (da tabela `default.creators_scrape_wiki`).
- Exemplo de um registro: `{'user_id': 'felipeneto', 'wiki_page': 'Felipe_Neto'}`.

- Crie o notebook "4 - analyze_creators" para analisar os creators.
  - Utilize o join das tabelas `default.users_yt` e `default.posts_creator`.
  - Mostre o top 3 posts ordenados por likes de cada creator nos últimos 6 meses (`user_id`, `title`, `likes`, `rank`).
  - Mostre o top 3 posts ordenados por views de cada creator nos últimos 6 meses (`user_id`, `title`, `views`, `rank`).
  - Mostre os `yt_user` que estão na tabela `default.post_creator` mas não estão na tabela `default.users_yt`.
  - Mostre a quantidade de publicações por mês de cada creator.
    - Exercício Extra 1: Mostre 0 nos meses que não têm vídeo.
    - Exercício Extra 2: Transforme a tabela no formato onde a primeira coluna é o `user_id` e há uma coluna para cada mês.

Parte 2: Desenho e Documentação de um Pipeline de Dados para Creators e Posts

Nesta parte, você projetará e documentará uma arquitetura de dados escalável e robusta para a ingestão e atualização contínua de dados de criadores e suas postagens, **considerando que você não possui os arquivos JSON do exercício anterior**.

Objetivo: Projetar um pipeline que consiga coletar e atualizar continuamente dados de criadores e posts, partindo do zero em relação aos dados iniciais.

Requisitos da Arquitetura:

- **Orquestrador:** Qual orquestrador você utilizaria e por quê?
- **Modelagem de Dados:** Diagrama de relacionamento e documentação das tabelas

(incluindo `creators`, `posts`, etc.).

- **Extração de Dados:** Como você faria a extração inicial e as atualizações desses dados (APIs, web scraping)?
- **Etapas do Pipeline:** Descreva o fluxo de dados (extração, transformação, carga, etc.).
- **Monitoramento e Qualidade**: Como você monitoraria o pipeline e avaliaria a qualidade dos dados finais?
- **Boas Práticas:** Qual fluxo e boas práticas de engenharia de software (Gitflow, princípios, etc.) seriam importantes?

Entrega: A documentação e o diagrama da arquitetura devem ser incluídos no seu projeto GitHub final.-----Essa versão enfatiza a restrição de não ter os arquivos JSON, direcionando o foco para o design de um pipeline capaz de lidar com a ingestão e atualização contínua dos dados a partir das fontes originais.