Detalhes do Checkpoint 3

Modelagem e Orquestração

Objetivo

Nesta etapa do desafio, o propósito é transformar e modelar os dados da **Adventure Works**, que já foram carregados e estão disponíveis no ambiente do **Databricks**, utilizando o **dbt**. O objetivo é construir um pipeline de dados completo e orquestrado, desde a **camada staging** até a criação do **modelo dimensional** na **camada marts**. Ao final, o pipeline deverá ser capaz de popular um **Data Warehouse** que responda às perguntas de negócio da área de vendas (Sales) da **AW**.

Diretrizes do Checkpoint

Para alcançar o objetivo deste checkpoint, é fundamental seguir uma abordagem estruturada. Recomendamos que você siga as etapas abaixo como um guia para o desenvolvimento, adaptando-as conforme a necessidade do projeto:

1. **Estruturação do projeto dbt:** Comece configurando seu projeto **dbt**. Organize as pastas para as camadas. Utilize um arquivo **dbt_project.yml** para definir as configurações globais, como caminhos de modelos e materializações.
2. **Modelagem da camada staging:** Crie modelos na camada staging para cada uma das fontes de dados ingeridas. O objetivo desta camada é limpar, padronizar e renomear colunas, além de realizar transformações de tipos de dados. Use prefixos para identificar esses modelos.
3. **Desenvolvimento da camada intermediate (opcional):** Se necessário, utilize a camada intermediate para criar modelos que unem ou pré-agregram dados da camada staging.
4. **Construção da camada marts:** Desenvolva os modelos finais na camada marts. Aqui, você construirá as tabelas de fatos e dimensões que responderão diretamente às perguntas de negócio. Pense em como os analistas e stakeholders da **AW** irão consumir esses dados.
5. **Documentação e testes:** À medida que desenvolve os modelos, crie a documentação e os testes de qualidade.
6. **Orquestração do Pipeline:** Com os modelos **dbt** prontos, desenvolva a orquestração. Crie **DAGs** para executar todo o fluxo, desde a ingestão até a

execução dos comandos para criação dos modelos e execução dos testes.

7. **Conteinerização:** Utilize **Docker** (ou outra ferramenta) para conteinerizar sua aplicação de orquestração, garantindo que o ambiente seja consistente e facilmente replicável.
8. **Versionamento:** Utilize o **Git** de forma consistente durante todo o processo. Faça **commits** com mensagens claras para registrar o progresso e as alterações no projeto.

Entrega

Para este checkpoint, você deverá entregar os seguintes itens:

- **Apresentação de Slides:**
  - Apresentação clara e concisa da solução.
  - Detalhamento do fluxo de dados desde as fontes até a construção do DW.
  - Justificativa das principais decisões técnicas tomadas
  - O uso de diagramas que ilustrem a arquitetura e o fluxo de dados é recomendado.

- **Repositório no GitHub/Bitbucket:**
  - Compartilhamento do link para um repositório Git privado, concedendo acesso ao seu avaliador (o acesso também deverá ser concedido a membros do L&D, caso seja requisitado).
  - O repositório deve conter todos os arquivos da solução de ingestão, modelagem e orquestração (scripts, notebooks, arquivos de configuração, etc.).
  - Um arquivo README.md detalhado, contendo:
    - Descrição do projeto e da solução de ingestão.
    - Instruções claras para configurar o ambiente de desenvolvimento (se houver dependências específicas).
    - Passo a passo para executar o pipeline completo (ingestão, modelagem e orquestração).
    - Listagem de todas as dependências do projeto e como instalá-las.
  - O código deve estar bem organizado, comentado e seguir boas práticas de desenvolvimento.

- **Documentação do Data Warehouse:**
  - Acesso à documentação gerada pelo dbt (via dbt docs serve ou via arquivos).

Observações adicionais

- Configure e use o seu próprio ambiente do Databricks Free Edition.
- Configure o catálogo e esquemas para organizar as tabelas em camadas.
- Os dados brutos já foram carregados no ambiente do Databricks no checkpoint anterior.
- Realize alterações no seu projeto, conforme feedbacks do avaliador do checkpoint 2.
- O dbt deve ser a principal ferramenta de transformação e modelagem.
- A escolha da ferramenta de orquestração é livre, mas a solução deve ser conteinerizada.
- Não altere o projeto no repositório remoto após o prazo final, sob pena de penalização na nota.
- Envie o repositório compactado em ZIP por e-mail, juntamente com a apresentação. Atente-se às variáveis de ambiente.
- Não exponha dados sensíveis ou credenciais no código ou no repositório.
- A apresentação não deve exceder 40 minutos, sob pena de penalização na nota.
- Não hesite em tirar dúvidas sobre os requisitos do desafio.
- Garanta que seu pipeline seja idempotente, ou seja, que reexecuções da mesma DAG não gerem dados duplicados ou inconsistentes.
- Lembre-se que o objetivo final é gerar valor para a Adventure Works. Todas as decisões de modelagem e arquitetura devem ser guiadas pelas necessidades de negócio.
- A dupla da Jornada de Negócios não deverá auxiliar na criação/construção do projeto e da apresentação.
- Durante a apresentação ao avaliador, a dupla da Jornada de Negócios deverá participar apenas como ouvinte.
- SQL Style Guide: Clique aqui.
- dbt Coding Conventions: Clique aqui.
- Code Style Python: Clique aqui.

Critérios de Avaliação

1. Modelagem de Dados:

a. Aplicação correta das boas práticas do dbt, com clara separação das camadas staging, intermediate (opcional) e marts.
b. Uso de CTEs para organizar e modularizar a lógica dentro dos modelos.

c. Nomenclatura clara e consistente dos modelos, utilizando prefixos para a camada de staging e para a camada de marts.
d. Documentação completa de todos os modelos e fontes de dados em arquivos YAML individuais, descrevendo tabelas e colunas.
e. Configuração de testes genéricos (ex: unique, not_null, relationships) para garantir a integridade dos dados.
f. Construção de pelo menos 2 testes singulares.
g. Modelos da camada marts materializados como tabelas.
h. Construção de ao menos uma tabela fato adicional que agregue valor à análise, como uma Fato Snapshot Período, Fato Snapshot Acumulado ou uma Fato Agregada.
i. O modelo de dados final deve ser capaz de atender de forma clara às necessidades de análise da Adventure Works.

2. Orquestração do Pipeline:

a. A orquestração deve cobrir todo o ciclo de vida dos dados, desde a execução dos scripts de ingestão até a execução dos modelos na camada marts e aplicação dos testes.
b. Integração com tecnologias de conteinerização (ex: Docker) para encapsular o ambiente de orquestração.
c. As DAGs devem ser bem estruturadas, com tarefas claras, dependências lógicas e de fácil compreensão.
d. Configuração de um agendamento diário para a execução de todo o pipeline.
e. O pipeline deve ser idempotente.
f. Implementação de tratamento de erros, retentativas e alertas básicos para monitorar o pipeline.

3. Organização, Documentação e Qualidade:

a. O código SQL dos modelos dbt e códigos usados na orquestração devem ser claros, bem comentados e seguir padrões de estilo.
b. Aplicação de boas práticas de desenvolvimento, como DRY (Don't Repeat Yourself).
c. Uso adequado de Git, com um histórico de commits, mensagens descritivas e uso de branches.
d. Gerenciamento adequado de configurações e segredos, evitando a exposição de informações sensíveis no código (ex: variáveis de ambiente).
e. O README.md deve ser completo, claro e fornecer todas as informações necessárias para outro desenvolvedor entender e executar o projeto (descrição do projeto, instruções de configuração e execução, listagem de dependências).
f. A estrutura de arquivos e pastas do repositório deve ser lógica e organizada.
g. A documentação do DW (dbt docs) deve estar disponível e ser facilmente acessível.

4. Entrega, Apresentação e Comunicação:

a. Compartilhamento correto do repositório GitHub (privado, com acesso ao avaliador).
b. A apresentação da solução deve ser clara, concisa e focada nos resultados e decisões importantes. O uso de fluxos e diagramas para ilustrar a arquitetura é recomendado.
c. Capacidade de justificar as escolhas técnicas feitas ao longo do projeto.
d. Habilidade de explicar conceitos técnicos de Analytics Engineering de forma clara.
e. Demonstração prática e funcional da estrutura do projeto, das DAGs de orquestração e dos modelos finais (ao vivo).
f. Capacidade de responder a perguntas do avaliador, demonstrando um entendimento profundo do projeto, dos dados e da solução.