Case de Engenharia de Dados: Construção de uma Arquitetura Lake House para Administradoras de Condomínio e Imobiliárias

Contexto:

Você foi contratado como engenheiro de dados para uma empresa fictícia chamada **CondoManage**, que fornece serviços de administração de condomínios e imobiliárias. A empresa está migrando sua infraestrutura de dados para uma arquitetura de Lake House para melhor gerenciar e analisar grandes volumes de dados de propriedades, moradores e transações.

Objetivo:

Seu objetivo é construir uma arquitetura de Lake House moderna, desenvolver processos de ingestão de dados e implementar transformações de dados usando Spark e Python.

Dados Disponíveis:

1. condominios (tabela postgresql):
- `condominio_id`: Identificador único do condomínio
- `nome`: Nome do condomínio
- `endereco`: Endereço do condomínio

```
CREATE TABLE condominios (
    condominio_id SERIAL PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    endereco TEXT NOT NULL
);
```

2. moradores (tabela postgresql):
- `morador_id`: Identificador único do morador

- `nome`: Nome do morador
- `condominio_id`: Identificador do condomínio onde mora
- `data_registro`: Data de registro do morador

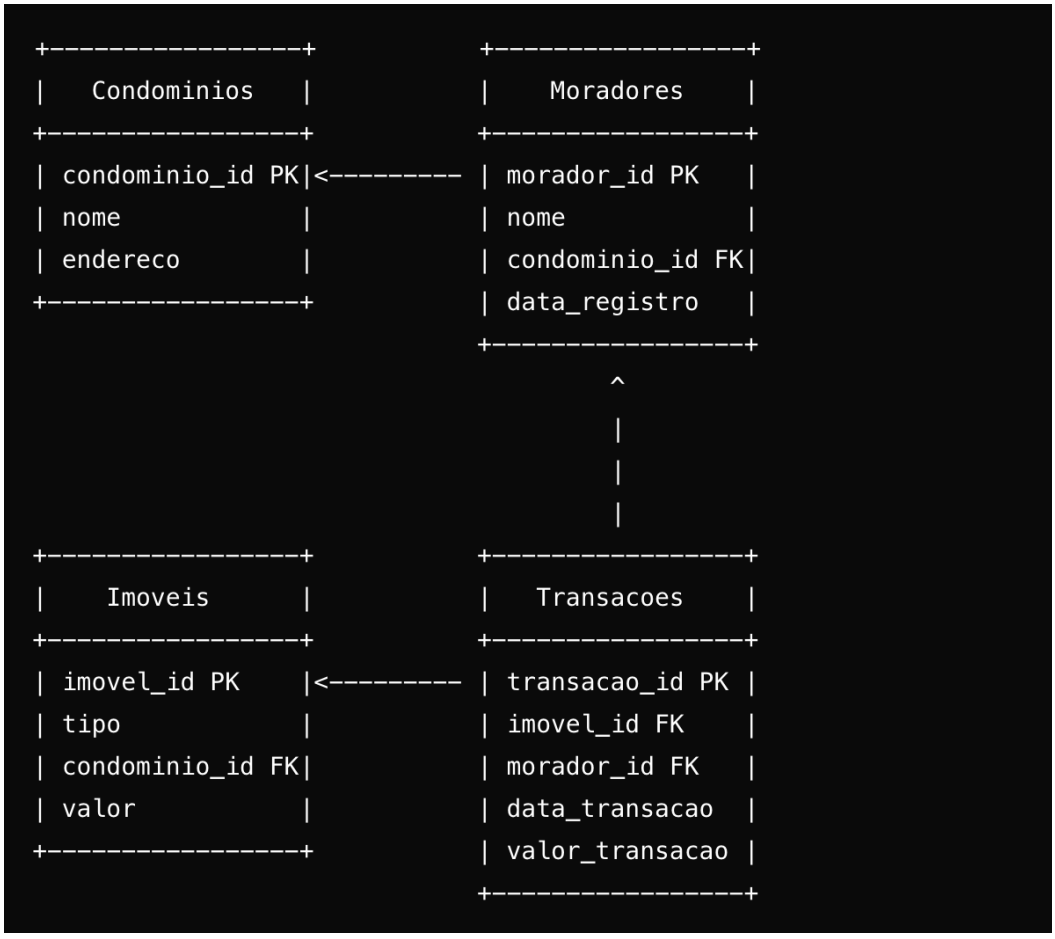
```
CREATE TABLE moradores (
    morador_id SERIAL PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    condominio_id INT NOT NULL,
    data_registro DATE NOT NULL,
    FOREIGN KEY (condominio_id) REFERENCES
condominios(condominio_id)
);
```

3. imoveis (tabela postgresql):
- `imovel_id`: Identificador único do imóvel
- `tipo`: Tipo de imóvel (ex: apartamento, casa)
- `condominio_id`: Identificador do condomínio onde o imóvel está localizado
- `valor`: Valor do imóvel

```
CREATE TABLE imoveis (
    imovel_id SERIAL PRIMARY KEY,
    tipo VARCHAR(50) NOT NULL,
    condominio_id INT NOT NULL,
    valor NUMERIC(15, 2) NOT NULL,
    FOREIGN KEY (condominio_id) REFERENCES
condominios(condominio_id)
);
```

4. Transacoes (tabela postgresql):
- `transacao_id`: Identificador único da transação
- `imovel_id`: Identificador do imóvel transacionado
- `morador_id`: Identificador do morador que realizou a transação
- `data_transacao`: Data da transação
- `valor_transacao`: Valor da transação

```
CREATE TABLE transacoes (
    transacao_id SERIAL PRIMARY KEY,
    imovel_id INT NOT NULL,
    morador_id INT NOT NULL,
    data_transacao DATE NOT NULL,
    valor_transacao NUMERIC(15, 2) NOT NULL,
    FOREIGN KEY (imovel_id) REFERENCES imoveis(imovel_id),
    FOREIGN KEY (morador_id) REFERENCES moradores(morador_id)
);
```



Tarefas:

1. Arquitetura Lake House:

- Desenhe uma arquitetura de Lake House para a CondoManage que minimamente inclua as seguintes camadas:
  - **Data Ingestion Layer:** Ingestão de dados de várias fontes (arquivos CSV, APIs, etc.).
  - **Data Storage Layer:** Armazenamento de dados brutos e processados em um Data Lake.
  - **Data Processing Layer:** Processamento de dados usando Spark.
  - **Data Serving Layer:** Exposição de dados processados para análise.

2. Ingestão de Dados:

- Escreva um script em Python para carregar os dados de `condomínios`, `moradores`, `imóveis.csv` e `transações` no Data Lake, pode usar como padrão um banco de dados postgreSQL
- Utilize o Apache Spark para processar e validar os dados durante a ingestão.

3. Transformação de Dados:

- Usando Spark, crie scripts para:
  - Calcular o total de transações por condomínio.
  - Calcular o valor total das transações por morador.
  - Agregar as transações diárias por tipo de imóvel.
- Salve os dados transformados no Data Lake em um formato otimizado (como Parquet ou ORC).

4. Data Processing Pipelines:

- Implemente pipelines de processamento contínuo (streaming) para ingestão de dados em tempo real, utilizando Spark Structured Streaming.
- Garanta a escalabilidade e a tolerância a falhas dos pipelines.

5. Documentação e Monitoramento:

- Descreva como você documentaria o design e a implementação da arquitetura e dos pipelines.

- Descreva como você montaria uma camada de monitoramento e alertas para os pipelines de dados usando ferramentas como Apache Kafka, Apache Airflow, ou AWS Glue.