📂 Case Técnico: Data Engineer - CRM Data

1. Introdução e Cenário

Você está assumindo a posição de Analytics Engineer em nossa empresa. Atualmente, o time de Marketing utiliza Salesforce Marketing Cloud (Email) e um parceiro externo para WhatsApp. Embora tenhamos dados de disparo e dados de conversão, existe um "buraco negro" no meio do funil.

O time de Engenharia de Dados fornece os logs brutos ("Raw Data") via ingestão de API, mas esses dados chegam com **baixa qualidade, sem padronização de chaves e com fusos horários distintos**.

Seu Objetivo: Desenhar e implementar um pipeline de modelagem que saneie esses dados, unifique a visão do cliente e aplique uma regra de atribuição de conversão robusta para alimentar os dashboards executivos.

---

2. Os Dados (Input)

Você receberá 3 arquivos CSV simulando tabelas do nosso Data Lake (BigQuery).

1. **`raw_sfmc_email_logs.csv`**: Logs de interação de Email.
    - *Nota Técnica:* O servidor de disparo está configurado em **UTC**. O campo `message_details` é uma string contendo um JSON com metadados variáveis.
2. **`raw_whatsapp_provider.csv`**: Logs de interação de WhatsApp.
    - *Nota Técnica:* O provedor brasileiro envia os logs em **Horário de Brasília (BRT)**. A identificação da campanha vem "suja" dentro de uma tag de texto.
3. **`crm_user_base.csv`**: Tabela dimensional de usuários e conversões.
    - *Nota Técnica:* Única tabela que relaciona Email com Telefone. Contém a data de conversão (contratação do cartão).

---

3. O Desafio (Escopo de Trabalho)

O case deve ser entregue cobrindo os 4 pilares abaixo:

Pilar A: Engenharia e Limpeza (SQL / BigQuery Dialect)

Os dados não conversam entre si nativamente. Você precisa criar uma camada de preparação (`staging`).

1. **Tratamento de Chaves:** Normalize emails e telefones para garantir que o `JOIN` entre as tabelas de canais e a base de usuários não tenha perda de dados devido a formatação (ex: `+55 (11)` vs `5511`).
2. **Harmonização Temporal:** Converta todos os timestamps para um fuso horário único (**America/Sao_Paulo**) antes de qualquer cálculo.
3. **Parsing de Dados:** Extraia o código da campanha (`campaign_code`) limpo, retirando-o do JSON no Email e limpando as strings no WhatsApp.

Pilar B: Modelagem de Negócio e Atribuição (SQL)

Crie uma tabela final (`fct_attribution`) que responda: *"Qual campanha gerou esta conversão?"*. **Regra de Negócio (Weighted Last Touch):**

1. **Janela de Olhares:** Apenas interações ocorridas nos **7 dias** anteriores à conversão são elegíveis.
2. **Conflito de Canais (A "Pegadinha"):** Se um usuário interagir com múltiplos canais no **mesmo dia**, a prioridade **não** é definida puramente pelo horário, mas pelo "peso" da interação:
    - Peso 1 (Maior): **WhatsApp Read**
    - Peso 2 (Médio): **Email Click**
    - Peso 3 (Menor): **Email Open**
    - *Exemplo:* Se o usuário abriu um Email às 20:00 e leu um Whats às 09:00 do mesmo dia, o **WhatsApp** vence a atribuição.
    - Interações de apenas "sent" ou "delivered" não geram atribuição, a menos que sejam a única interação no período.

Pilar C: Pipeline e Automação (Python)

Imagine que este processo deve rodar diariamente no Airflow.

1. Escreva um script **Python** que simule a ingestão do arquivo `raw_sfmc_email_logs.csv`.
2. **Quality Gate:** O script deve conter uma função de validação que identifique linhas onde o campo JSON (`message_details`) está mal formado/quebrado e **impeça** a carga dessas linhas ruins, logando o erro.

Pilar D: Arquitetura e Governança

No seu `README.md` ou em um diagrama à parte:

1. **Desenho da Solução:** Como você organizaria as camadas no BigQuery (ex: Raw -> Bronze -> Silver -> Gold)? Onde entrariam as procedures?
2. **Backfill:** Explique como sua query ou procedure lidaria com uma reprocessamento de dados passados (ex: se descobrirmos um erro na regra de atribuição e precisarmos rodar o último mês de novo).

---

4. Entregáveis

Esperamos receber um link de repositório (GitHub/GitLab) ou arquivo ZIP contendo:

- **Códigos SQL:** Scripts separados para criação das tabelas de staging e a tabela final de atribuição.
- **Código Python:** Script de ingestão com validação de JSON.
- **Documentação:** Um arquivo `README.md` explicando suas decisões técnicas, suposições assumidas e as respostas do Pilar D.

Arquivos de dados serão enviados em anexo ao e-mail.