**Relatório Técnico - Tratamento de Dados**

**1. Objetivo**

O objetivo deste projeto foi construir um pipeline de ingestão, padronização e integração de dados brutos oriundos de três fontes distintas (customers.json, orders.csv e web\_events.csv), gerando tabelas limpas e coerentes para posterior análise, visualização e carga em bancos de dados relacionais.

**2. Ferramentas Utilizadas**

* **Python 3.x**: linguagem principal para manipulação de dados.
* **Pandas**: biblioteca para leitura, limpeza e integração dos dados.
* **JSON**: formato de entrada para dados estruturados de clientes.
* **Unidecode**: biblioteca usada para remover acentuação dos nomes de estados.
* **Datetime**: manipulação de datas com múltiplos formatos.
* **CSV**: formato de saída das tabelas tratadas.
* **Draw**.io:
* **Github:**

**3. Descrição do Pipeline**

**a) Leitura e unificação dos dados**

* Leitura de arquivos .csv e .json com suporte a diferentes codificações e formatações.
* Conversão inicial das colunas de data com formatos múltiplos (ISO, brasileiro, etc.).

**b) Tratamento da tabela customers**

* Padronização de e-mails para letras minúsculas.
* Remoção de duplicidades por e-mail, mantendo o registro mais recente.
* Conversão da coluna registration\_date para múltiplos formatos.
* Mapeamento de estados para siglas padronizadas, incluindo variações com e sem acento.
* Criação de uma tabela de referência states com state\_id, state\_name, state\_code.
* Integração de state\_id via merge com a tabela customers.

**c) Tratamento da tabela orders**

* Conversão da coluna order\_date com a mesma lógica aplicada a customers.
* Padronização de valores monetários, transformando 1.234,56 ou 1,234.56 em float.
* Eliminação de pedidos duplicados com base no order\_id.

**d) Tratamento da tabela web\_events**

* Conversão de event\_timestamp para datetime real, respeitando formato ISO quando necessário.
* Remoção de espaços desnecessários em user\_agent.
* Extração e expansão da coluna metadata (formato JSON) para colunas auxiliares.
* Eliminação de duplicidade por event\_id.
* Integração com customers via user\_email.

**e) Exportação dos resultados**

* Tabelas exportadas no formato .csv, com datas no padrão DD/MM/AAAA:
  + customers\_clean.csv
  + orders\_clean.csv
  + web\_events\_clean.csv
  + states.csv

**4. Principais Desafios e Soluções**

* Desafios
  1. Datas em formatos mistos (ISO, brasileiro com ou sem barras);
  2. Estados com variações (com acento, sem acento e abreviados);
  3. Duplicidade de clientes por e-mail;
  4. Campos numéricos como string com virgula;
  5. Coluna metadata como JSON dentro de CSV.
* Solução
  1. Criação de função de parsing flexível com múltiplos formatos (%d/%m/%Y, %Y-%m-%d, etc.);
  2. Uso de unidecode + map padronizado + merge com tabela *states;*
  3. Conversão de e-mails para lowercase e ordenação por registration\_date para manter o mais recente;
  4. Substituição controlada e conversão para float com str.replace (',', '.').
  5. Expansão com json.loads() e apply(pd.series) para extrair atributos.

1. **Disponibilização da Pipeline**

Os dados gerados na análise estão disponíveis e podem ser consultados em:

https://github.com/FabioFeSantos/TesteGallagher/

1. **Considerações Finais**

O pipeline desenvolvido garante consistência e integridade referencial entre as tabelas, com formatação compatível com padrão ideal para relatórios e bancos de dados. Trazendo escalabilidade para aceitar novos registros com o mesmo modelo de dados.