PROJETO APLICADO DE PIPELINE DE DADOS - FASE 02

Disciplina: Pipeline de Dados - IA-CD

VISÃO GERAL DA FASE 02

Na Fase 01, vocês escolheram um tema, exploraram os dados e criaram o planejamento inicial. Agora é hora de estruturar melhor o pipeline, implementando transformações de dados, criando uma organização em camadas e integrando com banco de dados.

O que vocês farão nesta fase:

- Organizar dados em camadas (Bronze, Silver, Gold)
- Implementar transformações e limpeza de dados
- Criar validações básicas de qualidade
- Integrar com SQLite
- Criar modelo de dados relacional
- Documentar o processo ETL

CONCEITOS IMPORTANTES

1. Arquitetura em Camadas (Bronze, Silver, Gold)

Esta é uma forma de organizar dados que facilita manutenção e evolução do pipeline:

```
BRONZE (Dados Brutos)
  - Dados exatamente como chegam da fonte
  - Sem transformações
  - Serve como backup histórico

SILVER (Dados Limpos)
  - Dados validados e limpos
  - Tipos de dados corretos
  - Valores nulos tratados
  - Duplicatas removidas

GOLD (Dados Agregados)
  - Métricas e agregações
  - Dados prontos para análise
  - Otimizados para consultas
```

2. ETL - Extract, Transform, Load

Extract (Extração): Ler dados da fonte original
Transform (Transformação): Limpar, validar e preparar os dados
Load (Carga): Salvar no destino final (banco de dados)

3. Qualidade de Dados

Aspectos importantes a validar:

- **Completude:** Existem valores faltantes?
- **Consistência:** Os dados fazem sentido?
- **Unicidade:** Existem duplicatas?
- **Validade:** Os valores estão dentro do esperado?

ENTREGÁVEIS DA FASE 02

ENTREGÁVEL 1: Notebooks de Transformação

Criem 3 notebooks Jupyter organizados:

Notebook 1: Bronze Layer (01_bronze_layer.ipynb)

```
import pandas as pd
import os
from datetime import datetime

# Criar estrutura de pastas
os.makedirs('data/bronze', exist_ok=True)
os.makedirs('data/silver', exist_ok=True)
os.makedirs('data/gold', exist_ok=True)

# Carregar dados da fonte original
df_raw = pd.read_csv('data/source/seu_dataset.csv')

print(f"Dados carregados: {df_raw.shape[0]} linhas, {df_raw.shape[1]} colunas")

# Adicionar informações de quando os dados foram carregados
df_raw['data_ingestao'] = datetime.now()
df_raw['fonte_arquivo'] = 'seu_dataset.csv'

# Salvar na camada Bronze (sem modificar os dados originais)
df_raw.to_csv('data/bronze/dados_brutos.csv', index=False)
print("Dados salvos na camada Bronze")

# Visualizar primeiras linhas
df_raw.head()
```

Notebook 2: Silver Layer (02_silver_layer.ipynb)

```
import pandas as pd
import numpy as np

# Carregar dados da camada Bronze
df = pd.read_csv('data/bronze/dados_brutos.csv')
print(f"Dados originais: {df.shape}")

# ========================================
# TRANSFORMAÇÃO 1: Remover Duplicatas
# ========================================
linhas_antes = len(df)
df = df.drop_duplicates()
linhas_depois = len(df)
print(f"Duplicatas removidas: {linhas_antes - linhas_depois}")

# ========================================
# TRANSFORMAÇÃO 2: Tratar Valores Nulos
# ========================================

# Analisar valores nulos
print("\nValores nulos por coluna:")
print(df.isnull().sum())

# Exemplo: preencher valores numéricos com a mediana
colunas_numericas = df.select_dtypes(include=[np.number]).columns
for coluna in colunas_numericas:
    if df[coluna].isnull().any():
        mediana = df[coluna].median()
        df[coluna].fillna(mediana, inplace=True)
        print(f"Coluna {coluna}: preenchida com mediana = {mediana}")

# Exemplo: preencher valores categóricos com 'Desconhecido'
colunas_texto = df.select_dtypes(include=['object']).columns
for coluna in colunas_texto:
    if df[coluna].isnull().any():
        df[coluna].fillna('Desconhecido', inplace=True)
        print(f"Coluna {coluna}: preenchida com 'Desconhecido'")
```

```
# ========================================
# TRANSFORMAÇÃO 3: Converter Tipos de Dados
# ========================================

# Exemplo: converter colunas de data
# df['data_compra'] = pd.to_datetime(df['data_compra'])

# Exemplo: converter para categorias
# df['categoria'] = df['categoria'].astype('category')

# ========================================
# TRANSFORMAÇÃO 4: Padronizar Valores
# ========================================

# Exemplo: padronizar texto (remover espaços, colocar em maiúsculo)
# for coluna in ['nome', 'cidade', 'categoria']:
#     if coluna in df.columns:
#         df[coluna] = df[coluna].str.strip().str.upper()

# ========================================
# TRANSFORMAÇÃO 5: Remover Outliers
# ========================================

# Exemplo: remover outliers usando IQR para colunas numéricas
# def remover_outliers(df, coluna):
#     Q1 = df[coluna].quantile(0.25)
#     Q3 = df[coluna].quantile(0.75)
#     IQR = Q3 - Q1
#     limite_inferior = Q1 - 1.5 * IQR
#     limite_superior = Q3 + 1.5 * IQR
#     return df[(df[coluna] >= limite_inferior) & (df[coluna] <= limite_superior)]

# df = remover_outliers(df, 'valor_compra')

# ========================================
# VALIDAÇÕES BÁSICAS
# ========================================

print("\n=== VALIDAÇÕES DE QUALIDADE ===")

# Completude
total_celulas = df.shape[0] * df.shape[1]
celulas_preenchidas = df.count().sum()
completude = (celulas_preenchidas / total_celulas) * 100
print(f"Completude: {completude:.2f}%")

# Unicidade
duplicatas = df.duplicated().sum()
unicidade = ((len(df) - duplicatas) / len(df)) * 100
print(f"Unicidade: {unicidade:.2f}%")

# Adicionar informações de processamento
df['data_processamento'] = datetime.now()

# Salvar na camada Silver
df.to_csv('data/silver/dados_limpos.csv', index=False)
print(f"\nDados limpos salvos: {df.shape}")
```

Notebook 3: Gold Layer (03_gold_layer.ipynb)

```
import pandas as pd

# Carregar dados da camada Silver
df = pd.read_csv('data/silver/dados_limpos.csv')

# ========================================
# AGREGAÇÃO 1: Métricas Diárias
# ========================================

# Exemplo para e-commerce
# metricas_diarias = df.groupby('data_compra').agg({
#     'id_pedido': 'count',           # Total de pedidos
#     'valor_total': 'sum',           # Receita total
#     'id_cliente': 'nunique',        # Clientes únicos
#     'valor_total': 'mean'           # Ticket médio
# }).reset_index()
#
# metricas_diarias.columns = ['data', 'total_pedidos', 'receita_total',
#                             'clientes_unicos', 'ticket_medio']
#
# metricas_diarias.to_csv('data/gold/metricas_diarias.csv', index=False)

# ========================================
# AGREGAÇÃO 2: Análise de Clientes
# ========================================

# Exemplo: RFM (Recency, Frequency, Monetary)
# analise_clientes = df.groupby('id_cliente').agg({
#     'data_compra': 'max',           # Última compra (Recency)
#     'id_pedido': 'count',           # Frequência
#     'valor_total': 'sum'            # Valor monetário
# }).reset_index()
#
# analise_clientes.columns = ['id_cliente', 'ultima_compra',
#                             'frequencia', 'valor_total']
#
# # Calcular dias desde última compra
# analise_clientes['dias_ultima_compra'] = (
#     pd.to_datetime('today') - pd.to_datetime(analise_clientes['ultima_compra'])
# ).dt.days
#
# analise_clientes.to_csv('data/gold/analise_clientes.csv', index=False)

# ========================================
# AGREGAÇÃO 3: Desempenho de Produtos
# ========================================

# Exemplo
# produtos = df.groupby('produto').agg({
#     'id_pedido': 'count',           # Quantidade vendida
#     'valor_total': 'sum',           # Receita total
#     'id_cliente': 'nunique'         # Clientes únicos
# }).reset_index()
#
# produtos.columns = ['produto', 'quantidade_vendida', 'receita', 'clientes_unicos']
# produtos = produtos.sort_values('receita', ascending=False)
#
# produtos.to_csv('data/gold/desempenho_produtos.csv', index=False)

print("Agregações criadas e salvas na camada Gold")
```

ENTREGÁVEL 2: Integração com SQLite

Criem um notebook para carregar os dados no banco:

Notebook 4: Load to Database (04_load_database.ipynb)

```
import pandas as pd
import sqlite3
from datetime import datetime

# Conectar ao banco SQLite (cria se não existir)
conn = sqlite3.connect('data/pipeline.db')
print("Conexão com banco de dados estabelecida")

# ========================================
# CARREGAR DADOS LIMPOS (SILVER)
# ========================================

df_limpo = pd.read_csv('data/silver/dados_limpos.csv')

# Salvar em tabela principal
df_limpo.to_sql('tabela_principal', conn, if_exists='replace', index=False)
print(f"Tabela principal criada: {len(df_limpo)} registros")

# ========================================
# CARREGAR DADOS AGREGADOS (GOLD)
# ========================================

# Se você criou agregações, carregue-as também
# metricas = pd.read_csv('data/gold/metricas_diarias.csv')
# metricas.to_sql('metricas_diarias', conn, if_exists='replace', index=False)

# ========================================
# CRIAR TABELAS RELACIONADAS
# ========================================

# Exemplo: tabela de clientes
# clientes = df_limpo[['id_cliente', 'nome_cliente', 'email', 'cidade']].drop_duplicates()
# clientes.to_sql('clientes', conn, if_exists='replace', index=False)

# Exemplo: tabela de produtos
# produtos = df_limpo[['id_produto', 'nome_produto', 'categoria', 'preco']].drop_duplicates()
# produtos.to_sql('produtos', conn, if_exists='replace', index=False)

# ========================================
# VERIFICAR TABELAS CRIADAS
# ========================================

cursor = conn.cursor()
cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
tabelas = cursor.fetchall()
print("\nTabelas no banco de dados:")
for tabela in tabelas:
    print(f"  - {tabela[0]}")

conn.close()
print("\nDados carregados com sucesso no banco de dados!")
```

ENTREGÁVEL 3: Queries SQL para Análise

Criem um notebook com queries SQL para explorar os dados:

Notebook 5: SQL Queries (05_sql_queries.ipynb)

```
import pandas as pd
import sqlite3
```

```
import sqlite3

# Conectar ao banco
conn = sqlite3.connect('data/pipeline.db')

# ========================================
# QUERY 1: Visão Geral dos Dados
# ========================================

query = """
SELECT COUNT(*) as total_registros
FROM tabela_principal
"""
resultado = pd.read_sql_query(query, conn)
print("Total de registros:", resultado['total_registros'].values[0])

# ========================================
# QUERY 2: Top 10 (exemplo: produtos mais vendidos)
# ========================================

# query = """
# SELECT
#     produto,
#     COUNT(*) as quantidade_vendas,
#     SUM(valor_total) as receita_total
# FROM tabela_principal
# GROUP BY produto
# ORDER BY receita_total DESC
# LIMIT 10
# """
# top_produtos = pd.read_sql_query(query, conn)
# print("\nTop 10 Produtos:")
# print(top_produtos)

# ========================================
# QUERY 3: Análise Temporal
# ========================================

# query = """
# SELECT
#     DATE(data_compra) as data,
#     COUNT(*) as total_vendas,
#     SUM(valor_total) as receita
# FROM tabela_principal
# GROUP BY DATE(data_compra)
# ORDER BY data
# """
# vendas_diarias = pd.read_sql_query(query, conn)
# print("\nVendas por dia:")
# print(vendas_diarias.head())

# ========================================
# QUERY 4: Segmentação de Clientes
# ========================================

# query = """
# SELECT
#     cidade,
#     COUNT(DISTINCT id_cliente) as total_clientes,
#     AVG(valor_total) as ticket_medio
# FROM tabela_principal
# GROUP BY cidade
# ORDER BY total_clientes DESC
# """
# clientes_por_cidade = pd.read_sql_query(query, conn)
# print("\nClientes por cidade:")
```

```
# print(clientes_por_cidade)

conn.close()
```

ENTREGÁVEL 4: Relatório de Qualidade

Criem um notebook que analisa a qualidade dos dados:

Notebook 6: Data Quality Report (06_quality_report.ipynb)

```
 import pandas as pd
import matplotlib.pyplot as plt

# Carregar dados limpos
df = pd.read_csv('data/silver/dados_limpos.csv')

print("="*50)
print("RELATÓRIO DE QUALIDADE DE DADOS")
print("="*50)

# ========================================
# 1. COMPLETUDE
# ========================================

print("\n1. COMPLETUDE DOS DADOS")
print("-" * 50)

total_celulas = df.shape[0] * df.shape[1]
celulas_preenchidas = df.count().sum()
completude_geral = (celulas_preenchidas / total_celulas) * 100

print(f"Completude Geral: {completude_geral:.2f}%")
print("\nCompletude por coluna:")

for coluna in df.columns:
    valores_preenchidos = df[coluna].count()
    total = len(df)
    percentual = (valores_preenchidos / total) * 100
    print(f"  {coluna}: {percentual:.2f}%")

# ========================================
# 2. UNICIDADE
# ========================================

print("\n2. UNICIDADE DOS DADOS")
print("-" * 50)

duplicatas = df.duplicated().sum()
unicidade = ((len(df) - duplicatas) / len(df)) * 100
print(f"Linhas únicas: {unicidade:.2f}%")
print(f"Duplicatas encontradas: {duplicatas}")

# ========================================
# 3. CONSISTÊNCIA
# ========================================

print("\n3. CONSISTÊNCIA DOS DADOS")
print("-" * 50)

# Exemplo: verificar se valores numéricos são positivos
# for coluna in ['valor_total', 'quantidade']:
#     if coluna in df.columns:
#         negativos = (df[coluna] < 0).sum()
```

```
#         print(f"{coluna}: {negativos} valores negativos encontrados")

# Exemplo: verificar se datas estão no passado
# if 'data_compra' in df.columns:
#     df['data_compra'] = pd.to_datetime(df['data_compra'])
#     futuras = (df['data_compra'] > pd.Timestamp.now()).sum()
#     print(f"Datas no futuro: {futuras}")

# ========================================
# 4. VALIDADE
# ========================================

print("\n4. VALIDADE DOS DADOS")
print("-" * 50)

# Exemplo: verificar ranges válidos
# if 'idade' in df.columns:
#     fora_range = ((df['idade'] < 0) | (df['idade'] > 120)).sum()
#     print(f"Idades inválidas: {fora_range}")

# ========================================
# 5. VISUALIZAÇÃO
# ========================================

# Gráfico de completude
plt.figure(figsize=(10, 6))
completude_por_coluna = (df.count() / len(df) * 100).sort_values()
completude_por_coluna.plot(kind='barh')
plt.xlabel('Completude (%)')
plt.title('Completude dos Dados por Coluna')
plt.tight_layout()
plt.savefig('data/quality_report.png')
print("\nGráfico de qualidade salvo em: data/quality_report.png")

# ========================================
# SCORE FINAL
# ========================================

print("\n" + "="*50)
print("SCORE GERAL DE QUALIDADE")
print("="*50)

# Score simples: média de completude e unicidade
score_final = (completude_geral + unicidade) / 2
print(f"Score Final: {score_final:.2f}%")

if score_final >= 90:
    print("Classificação: EXCELENTE")
elif score_final >= 80:
    print("Classificação: BOM")
elif score_final >= 70:
    print("Classificação: REGULAR")
else:
    print("Classificação: NECESSITA MELHORIAS")
```

ENTREGÁVEL 5: Documentação

Atualizem o README.md do projeto:

```
# Pipeline de Dados - [Nome do Projeto]

## Descrição
[Breve descrição do projeto e problema que está resolvendo]

## Estrutura de Dados

### Camada Bronze
- Localização: `data/bronze/`
- Descrição: Dados brutos, sem transformações
- Arquivo: `dados_brutos.csv`

### Camada Silver
- Localização: `data/silver/`
- Descrição: Dados limpos e validados
- Arquivo: `dados_limpos.csv`
- Transformações aplicadas:
  1. Remoção de duplicatas
  2. Tratamento de valores nulos
  3. Conversão de tipos
  4. Padronização de valores
  5. Remoção de outliers

### Camada Gold
- Localização: `data/gold/`
- Descrição: Dados agregados para análise
- Arquivos:
  - `metricas_diarias.csv`
  - `analise_clientes.csv`
  - `desempenho_produtos.csv`

## Banco de Dados

- Tipo: SQLite
- Localização: `data/pipeline.db`
- Tabelas:
  - `tabela_principal`: Dados completos limpos
  - `clientes`: Informações de clientes
  - `produtos`: Catálogo de produtos
  - `metricas_diarias`: Agregações diárias

## Qualidade dos Dados

- Completude: XX%
- Unicidade: XX%
- Score Geral: XX%

## Como Executar

1. Execute os notebooks na ordem:
   - `01_bronze_layer.ipynb`
   - `02_silver_layer.ipynb`
   - `03_gold_layer.ipynb`
   - `04_load_database.ipynb`
   - `05_sql_queries.ipynb`
   - `06_quality_report.ipynb`

2. Consulte o banco de dados:
   ```python
   import sqlite3
   conn = sqlite3.connect('data/pipeline.db')
   # suas queries aqui
```
```

DICAS DO PROFESSOR

Sobre Transformações

- Comecem pelas transformações mais simples
- Testem cada transformação antes de passar para a próxima
- Mantenham sempre uma cópia dos dados originais no Bronze
- Documentem o "porquê" de cada transformação

Sobre Banco de Dados

- SQLite é suficiente para esta fase
- Não se preocupem com otimização ainda
- Foquem em ter os dados carregados corretamente
- Testem as queries antes de considerar concluído

Sobre Qualidade

- É normal ter dados imperfeitos
- Documentem os problemas encontrados
- Score de 70-80% já é um bom resultado
- O importante é entender os problemas e saber tratá-los

Trabalho em Equipe

- Dividam os notebooks entre os membros da equipe
- Revisem o código uns dos outros
- Mantenham comunicação constante
- Usem Git para versionamento

DATA DE ENTREGA

Prazo: 03/11/2025

Formato de Entrega:

- Arquivo ZIP com código completo (se preferir) - Abrirei uma atividade de entrega na D2L.
- Todos os notebooks devem executar sem erros

Bom trabalho na Fase 02!