FACULDADE DONADUZZI

2º PERÍODO IA E CD

Disciplina de Pipeline

E-COMMERCE ANALYTICS

Discente**:** Anthony Guilherme Mucelini, Gabrieli Cavalcante Boeira, Jhonatan da Silva Margraf, Luan Bitencourt Sarmento, Rafaela Eduarda de Oliveira Barreiros.

Docente: Wesley Antonio Santos de Andrade Sobreira

12 de Agosto de 2025 Toledo,
Paraná.

Sumário

1.Objetivo

O tema E-commerce Analytics foi escolhido por ser muito importante no mercado atual, já que cada vez mais pessoas compram pela internet. Analisar os dados das vendas ajuda as empresas a entender o comportamento dos clientes, melhorar as estratégias e aumentar as vendas.

Além disso, esse tema permite aplicar conhecimentos de tecnologia, estatística e inteligência artificial, tornando o projeto útil e atual para o mundo digital.

1.2 Definição do Problema

As lojas virtuais muitas vezes não entendem bem o comportamento dos seus clientes, o que dificulta planejar promoções, controlar o estoque e aumentar as vendas.

Importância desse problema:
Compreender como os clientes compram ajuda a empresa a tomar melhores decisões, criar ofertas mais atrativas e melhorar os resultados.

O impacto que esperamos alcançar com a análise dos dados, será possível entender os padrões de compra, prever demandas e criar dashboards que ajudem a visualizar as informações.

1.3 Dataset

Nome do dataset: E-commerce Data (Kaggle)

Descrição:
O conjunto de dados contém todas as transações realizadas entre 01/12/2010 e 09/12/2011 em uma loja online do Reino Unido, que vende presentes e produtos exclusivos. A loja não tem ponto físico e atende também clientes atacadistas.

Tamanho do dataset:
541.909 linhas × 8 colunas

Principais variáveis (colunas):

- **InvoiceNo:** número da fatura
- **StockCode:** código do produto
- **Description:** descrição do produto
- **Quantity:** quantidade vendida
- **InvoiceDate:** data da venda
- **UnitPrice:** preço unitário
- **CustomerID:** identificação do cliente
- **Country:** país do cliente

Problemas de qualidade encontrados:

- Coluna **Description:** 1.454 valores faltando (foram recuperados) ● Coluna **CustomerID:** 135.080 valores faltando (não recuperáveis).
  Mesmo assim, as linhas foram mantidas para não prejudicar análises sobre o volume total de vendas e valores.

1.4 Arquitetura Do Pipeline

Seguiremos com essas etapas:

1. **Coleta dos dados:** usar o dataset do Kaggle.
2. **Limpeza dos dados:** corrigir valores faltantes e ajustar formatos.
3. **Análise dos padrões:** identificar produtos mais vendidos, países com mais compras e comportamento dos clientes.
4. **Criação de dashboards:** visualizar os resultados de forma clara e interativa, mostrando informações como volume de vendas, lucro e frequência de compra.

1.5 Evolução Futura

Podemos evoluir o projeto com as seguintes melhorias:

- **Recomendações automáticas de produtos** com base nas compras anteriores.
- **Previsão de demanda**, ajudando a planejar o estoque.
- **Análises em tempo real**, para acompanhar as vendas no momento em que acontecem.

2. Estrutura de Dados

2.1 Camadas de Dados

Camada Bronze

- **Localização:** data/bronze/
- **Descrição:** Dados brutos, sem transformações
- **Arquivo:** dados_brutos.parquet

Camada Silver

- **Localização:** data/silver/
- **Descrição:** Dados limpos e validados
- **Arquivo:** dados_limpos.parquet
- **Transformações aplicadas:**
  1. Tratamento de valores nulos
  2. Análise de duplicatas (sem duplicatas)
  3. Análise de cancelamentos e valores negativos
  4. Análise de outliers (sem outliers)

Camada Gold

- **Localização:** data/gold/
- **Descrição:** Dados agregados, separados em Fatos e Dimensões para análise
- **Arquivos:**
    - dim_country.parquet
    - dim_date.parquet
    - dim_customer.parquet
    - dim_product.parquet
    - fact_all.parquet
    - rfm.parquet
    - most_purchased_products.parquet
    - metrics.parquet
    -

2.2 Banco de Dados

- **Tipo:** SQLite
- **Localização:** data/pipeline.db
- **Tabelas de Dimensão:**
    - country: Países
    - date: Datas de vendas
    - customer: Clientes registrados
    - product: Catálogo de produtos
    - fact: Notas fiscais (vendas, cancelamentos, taxas)

2.3 Método de Execução

Execute os notebooks na seguinte 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

2.4 Consulta ao Banco de Dados

```
import sqlite3

conn = sqlite3.connect('data/pipeline.db')

cursor = conn.cursor()
#Exemplo de Consulta
cursor.execute("SELECT * FROM customer LIMIT 10;")
result = cursor.fetchall()
print(result)

conn.close()
```

3.0 Querys:

Temos a Criação do Banco de dados:

Primeiro Passo: **Criar a tabela no banco a partir do CSV**
--Código--

```
import pandas as pd
import sqlite3

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

conn = sqlite3.connect('data/pipeline.db')

df.to_sql('dados_limpos', conn, if_exists='replace', index=False)

print("Tabelas existentes no banco:")
print(pd.read_sql_query("SELECT name FROM sqlite_master WHERE type='table';", conn))
##
```

1°**Quantos registros temos no banco**
-- Código--

```
query = """
SELECT COUNT(*) AS total_registros
FROM dados_limpos
"""
resultado = pd.read_sql_query(query, conn)
print("\nTotal de registros:", resultado['total_registros'].values[0])
```

2°**Top 10 produtos que mais vendem**
--Código—

```
query = """
SELECT
    Description AS produto,
    SUM(Quantity) AS quantidade_total_vendida,
    ROUND(SUM(Quantity * UnitPrice), 2) AS receita_total
FROM dados_limpos
WHERE Description IS NOT NULL
GROUP BY Description
ORDER BY receita_total DESC
LIMIT 10
"""
print("\nTop 10 Produtos Mais Vendidos:")
print(pd.read_sql_query(query, conn))
```

3°**Vendas por dia**

```
query = """
SELECT
    DATE(InvoiceDate) AS data,
    COUNT(DISTINCT InvoiceNo) AS total_vendas,
    ROUND(SUM(Quantity * UnitPrice), 2) AS receita_total
FROM dados_limpos
GROUP BY DATE(InvoiceDate)
ORDER BY data
"""
print("\nVendas por Dia:")
print(pd.read_sql_query(query, conn).head())
```

4° **Clientes por país**

--Código--

```
query = """
SELECT
    Country AS pais,
    COUNT(DISTINCT CustomerID) AS total_clientes,
    ROUND(AVG(Quantity * UnitPrice), 2) AS ticket_medio
FROM dados_limpos
WHERE CustomerID IS NOT NULL
GROUP BY Country
ORDER BY total_clientes DESC
"""
print("\nClientes por País:")
print(pd.read_sql_query(query, conn))
```

Encerramento: **Fechamos a conexão com o banco só no final.**

```
conn.close()
print("\nConexão encerrada").
```

4.0 Evolução Planejada do Banco de Dados (PostgreSQL/MySQL)

Embora o pipeline continue utilizando **PostgreSQL** como banco principal (pipeline.db), foi desenvolvida uma **extensão do projeto** preparando uma futura migração para bancos relacionais mais robustos, como **PostgreSQL** ou **MySQL**.
Essa evolução não substitui o SQLite atual; ela apenas documenta e implementa a **estrutura de um Data Warehouse**, caso desejado no futuro.

4.1 Objetivo da Evolução

O objetivo da evolução é possibilitar, futuramente: Utilizar um banco mais performático e escalável.
Transformar o pipeline em um **Data Warehouse real**, com Star Schema.
Suportar consultas analíticas mais pesadas.
Preparar a infraestrutura para dashboards em produção.
O PostegreSQL continuará sendo usado no pipeline, mas a estrutura futura já está pronta.

4.2 Criação do Data Warehouse (Star Schema)

O script inclui a criação das seguintes tabelas:
Tabelas Dimensionais
- dim_country
- dim_customer
- dim_date
- dim_product

Tabela Fato
- fact_sales

Tabela de Métricas
- metrics

Essas tabelas incluem:
- **Chaves primárias**
- **Chaves estrangeiras**
- **AUTO_INCREMENT**
- **Timestamps**
- **Índices para otimização**

Exemplo de índices criados:
- invoice_no
- stock_code
- customer_id
- country_id
- date_id
- total_price
- índice composto (invoice_no, date_id)

5.0 Execução Automática do Schema

O script executa:
1. Criação das tabelas dimensionais
2. Criação da tabela fato
3. Criação da tabela de métricas
4. Verificação das tabelas criadas
5. Visualização completa da estrutura das colunas

O PostgreSQL/MySQL só é utilizado **caso exista** e **a conexão funcione**.

5.1 Resumo

O projeto agora possui:

Pipeline completo em PostgreSQL funcionando normalmente.

Código adicional para migrar o pipeline para PostgreSQL/MySQL, incluindo:

criação de Data Warehouse completo,
star schema,
índices e constraints,
validação automática do schema.

6.0 Airflow

Para automatizar as etapas Bronze → Silver → Gold e garantir que o pipeline seja executado de forma programada e confiável, foi criado um fluxo no **Apache Airflow**.

Objetivo

O Airflow garante:
Execução automática do pipeline
Reprocessamento organizado
Histórico de execuções
Monitoramento visual das tarefas
Alertas em caso de falha

Estrutura do DAG

O DAG contém as seguintes tarefas:

1. **Camada de Bronze**
   Carrega o dataset bruto
   Armazena em data/bronze/
2. **Camada Silver**
   Realiza limpeza
   Remove nulos e inconsistências
   Salva em data/silver/
3. **Camada de Ouro**
   Cria agregações
   Gera tabelas fato e dimensões
   Salva em data/gold/
4. **Carrega os dados**
   Carrega dados no PostgreSQL (pipeline.db)

7.0 Monitoramento

Além do monitoramento básico implementado no pipeline, foi criado um mecanismo adicional para acompanhar a execução das camadas através de um arquivo de log gerado pelo Airflow. Esse log registra informações essenciais sobre cada execução do pipeline.

Cada linha contém:

data_execucao – horário em que a task rodou
camada – bronze, silver, gold, load_database ou quality
status – sucesso ou erro

tempo_segundos – quanto tempo a tarefa
levou
registros_processados – número de linhas tratadas em cada etapa
Esses dados permitem acompanhar a saúde e o desempenho do pipeline.

7.1 Métricas Utilizadas

1. Taxa de Sucesso
Percentual de execuções concluídas sem erro.
2. Tempo Médio por Camada
Média de segundos que cada camada levou para ser processada.

Visualizações Geradas
O monitoramento gera automaticamente dois gráficos:
Gráfico 1 – Status das Execuções
Exibe a quantidade de tarefas com sucesso ou erro.
Gráfico 2 – Tempo Médio por Camada
Mostra o desempenho de cada etapa do pipeline, facilitando a detecção de tarefas lentas.

Conclusão

O pipeline ETL funcionou bem, transformando dados brutos e cheios de inconsistências em tabelas organizadas e métricas prontas para análise. Cada etapa do processo ajudou a limpar, enriquecer e estruturar os dados, resultando em uma camada Gold confiável e fácil de usar nas análises.