Atividade Avaliativa Prática – Projeto de Pipeline de Dados (ETL Completo)

Descrição Geral da Atividade

Os alunos deverão desenvolver um pipeline completo de ETL (Extract, Transform, Load) integrando um banco NoSQL (MongoDB) a um ambiente de Data Warehouse em Oracle. O objetivo é simular um cenário real de um sistema de vendas, realizando ingestão de dados, tratamento, enriquecimento, modelagem dimensional e carga final em tabelas fato e dimensão.

🛠️ Escopo da Atividade

1. Extração (Extract)

- Ler documentos de vendas armazenados no MongoDB, contendo:
  - Dados do pedido
  - Dados do cliente
  - Itens comprado
  - Informações de pagamento
  - Informações de envio

2. Transformação (Transform)

Aplicar regras de qualidade e enriquecimento, incluindo:

- Normalização de strings (remoção de acentos, UPPER, TRIM)
- Conversão de datas de UTC → America/São Paulo
- Cálculo de métricas comerciais:
  - `gross_amount`
  - `net_amount`
  - `discount_percent`

- Tratamento das dimensões utilizando SCD Tipo 2:
  - Versões históricas de clientes
  - Versões históricas de produtos

3. Carga (Load)

Carregar os dados transformados em um Data Warehouse Oracle, populando:
Dimensões

- `DIM_CUSTOMER`

- `DIM_PRODUCT`
- `DIM_DATE`

Tabela Fato

- `FCT_SALES`, contendo medidas e chaves estrangeiras para as dimensões.

O carregamento deverá garantir integridade referencial.

📦 Componentes Obrigatórios do Projeto

1. Banco MongoDB com, no mínimo, 5.000 documentos de vendas.
2. Área de Staging (STAGING) no Oracle.
3. Dimensões e fato modeladas em Star Schema.
4. Pacote PL/SQL (DW.PKG_ETL_SALES) implementando:
    - normalização
    - conversão de datas
    - cálculos de métrica
    - UPSERT SCD2
    - carga final na fato
5. Registro de execução em tabela de log (`ETL_EXECUTION_LOG`).
6. Pipeline orquestrado via Python (script ou módulo).
7. Docker Compose contendo ao menos:
    - MongoDB
    - Oracle XE 11g

Contexto do Projeto – Pipeline de Dados para Domínio de Vendas (MongoDB → Oracle DW)

A empresa possui uma plataforma de e-commerce que processa milhares de pedidos diariamente. As informações operacionais são armazenadas em um banco **NoSQL** (MongoDB), devido à flexibilidade necessária para lidar com documentos de vendas altamente variáveis, com campos aninhados, estruturas semiesquematizadas e atualizações dinâmicas.

Embora o MongoDB sustente bem as operações transacionais e de front-end, a camada analítica da empresa ainda não possui um ambiente unificado e estruturado para geração de relatórios corporativos, painéis executivos, análise de rentabilidade, comportamento dos clientes e previsões de demanda. Além disso, as equipes de BI e Controladoria necessitam de dados consolidados, padronizados e confiáveis para alimentar indicadores financeiros e apoiar decisões estratégicas.

Para suprir essa necessidade, foi definido o desenvolvimento de um **pipeline de dados corporativo**, integrando a fonte NoSQL operacional ao **Data Warehouse em Oracle**, que servirá como repositório centralizado para consumo analítico. O pipeline deve ser totalmente automatizado, robusto, observável e resiliente a falhas.

O processo deverá contemplar:

🎯 Objetivos do pipeline

1. Ler as vendas do MongoDB (NoSQL).

2. Tratar e enriquecer:
    - Normalizar dados de cliente, produto, datas, meios de pagamento.
    - Calcular campos derivados (margem, descontos, flags de fraude etc.).
    - Resolver inconsistências (campos null, formatos diferentes).

3. Carregar dados tratados em um DW de vendas no Oracle, usando PL/SQL para:
    - Limpeza e enriquecimento final.
    - Upsert de dimensões (SCD simplificado).
    - Carga da tabela fato de vendas.

4. Python será o orquestrador de tudo.

Arquitetura do Pipeline (Anexo I)

Componentes principais:

- Fonte NoSQL: MongoDB (coleção `sales`).
- Camada de Staging no Oracle(schema STAGING - Anexo: II):
  - Tabela `stg_sales`.
- Camada de DW no Oracle (schema DW - Anexo: III):
  - `dim_customer`
  - `dim_product`
  - `dim_date`
  - `fct_sales`
- PL/SQL:
  - Pacote `pkg_etl_sales` para transformação, enriquecimento e carga(anexo: IV).
  - Triggers DW para geração de identificadores e auditoria de carga e operações na fato (Anexo V).
- Python (orquestração):
  - Script/serviço `etl_sales_pipeline.py`:
    - Conecta no MongoDB.
    - Lê vendas novas.
    - Grava na staging do Oracle.
    - Chama o pacote PL/SQL.
    - Faz logging e controle de execução.

3. Modelo de Dados – Exemplo

3.1. Documento NoSQL (MongoDB) – Coleção `sales`


```
{
  "_id": "6733fa10a1b2c34567890abc",
  "order_id": "ORD-2025-000123",
  "order_date": "2025-11-10T14:35:22Z",
  "customer": {
    "customer_id": "CUST-999",
    "name": "João Silva",
    "email": "joao.silva@example.com",
    "document": "12345678909"
  },
  "items": [
    {
      "product_id": "PROD-001",
      "product_name": "Notebook Gamer",
      "category": "Eletrônicos",
      "unit_price": 4500.00,
      "quantity": 1,
      "discount": 200.00
    },
    {
      "product_id": "PROD-002",
```

```
      "product_name": "Mouse Gamer",
      "category": "Acessórios",
      "unit_price": 150.00,
      "quantity": 2,
      "discount": 0.00
    }
  ],
  "payment": {
    "method": "CREDIT_CARD",
    "installments": 10,
    "status": "APPROVED"
  },
  "shipping": {
    "city": "Belo Horizonte",
    "state": "MG",
    "country": "BR",
    "shipping_value": 50.0
  }
}
```


4. Fluxo do Pipeline (Etapas)

4.1. Extração (Python → MongoDB)

- Conectar ao MongoDB.
- Buscar documentos de vendas **novas** ou **atualizadas** (ex.: filtro por `order_date` ou `_id` > último processado).
- Converter para estrutura tabular (flatten) – um registro por item de venda.

4.2. Carga na Staging (Python → Oracle)

- Conectar ao Oracle via `cx_Oracle` ou `oracledb`.
- Inserir em `stg_sales` (bulk insert).
- Registrar no **log de execução** (tabela `etl_execution_log`).

4.3. Transformação e Enriquecimento (PL/SQL no Oracle)

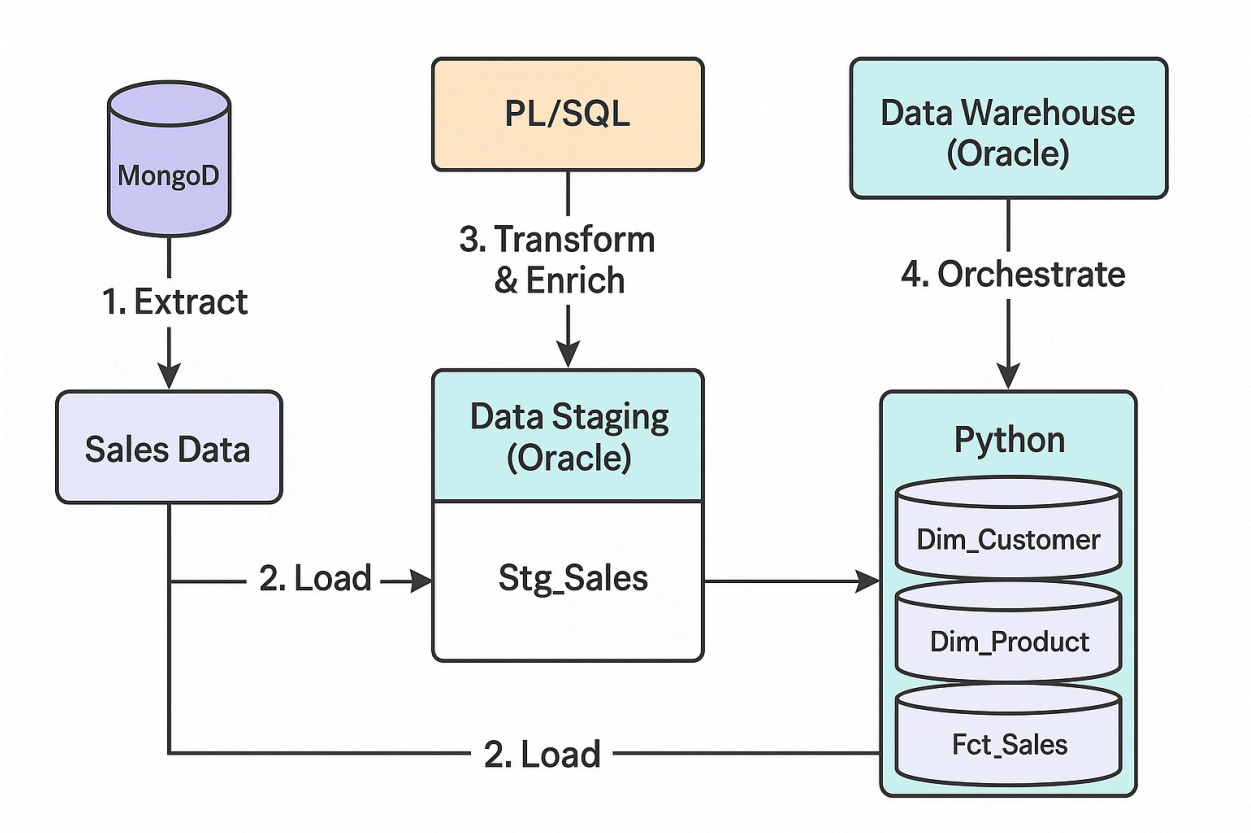
- PL/SQL roda em pacote `pkg_etl_sales`:
  - Normalização de campos (maiúsculas, trimming, conversão de datas).
  - Cálculo de métricas: `gross_amount = unit_price * quantity`, `net_amount = gross_amount - discount_value`.
  - Upsert de `dim_customer`, `dim_product`, `dim_date`.
  - Carga da `fct_sales`.

Python apenas chama os procedimentos PL/SQL.

4.4. Limpeza de Staging e Finalização

- PL/SQL ou Python limpa staging (dependendo da estratégia).
- Atualiza marca de “último `_id` processado” ou timestamp.
- Atualiza status de execução no log.

Anexo I - Arquitetura Geral



Anexo II - Staging - Dicionário de Dados

1. Tabela: STAGING.STG_SALES

Descrição: tabela de *staging* utilizada para armazenar dados brutos extraídos do MongoDB antes das transformações e carga no Data Warehouse.

Coluna	Tipo	Descrição e Finalidade
batch_id	NUMBER	Identificador do lote de carga. Permite rastrear execuções do pipeline.
id_mongo	VARCHAR2(50)	Chave `_id` do documento no MongoDB. Usada para controle de duplicidade e rastreamento da origem.
order_id	VARCHAR2(50)	Identificador do pedido na origem. Pode ser chave de negócio.
order_date_utc	TIMESTAMP	Data/hora do pedido na origem, em UTC. Usado no controle incremental e particionamento.

customer_id	VARCHAR2(50)	ID do cliente na origem. Usado no vínculo com a dimensão de clientes.
customer_name	VARCHAR2(200)	Nome do cliente. Pode vir com variações e será padronizado na camada DW.
customer_email	VARCHAR2(200)	E-mail do cliente, usado em análises de comportamento e marketing.
customer_document	VARCHAR2(20)	Documento do cliente (ex.: CPF), sem formatação. Pode ser validado via PL/SQL.
product_id	VARCHAR2(50)	Identificador do produto na origem. Usado no vínculo com a dimensão de produtos.
product_name	VARCHAR2(300)	Nome do produto no momento da venda (pode variar com o tempo).
product_category	VARCHAR2(200)	Categoria do produto no e-commerce. Útil para análises de portfólio.
quantity	NUMBER(10)	Quantidade vendida do item. Um pedido pode ter vários registros de itens.
unit_price	NUMBER(12,2)	Preço unitário do item no momento da compra.
discount_value	NUMBER(12,2)	Valor total de desconto aplicado ao item.
payment_method	VARCHAR2(50)	Método de pagamento (ex.: CARTÃO, PIX, BOLETO).
payment_status	VARCHAR2(50)	Status do pagamento (ex.: APROVADO, RECUSADO).
installments	NUMBER(3)	Quantidade de parcelas, quando aplicável.
shipping_city	VARCHAR2(100)	Cidade para onde o pedido foi enviado.
shipping_state	VARCHAR2(50)	UF/estado do destino.
shipping_country	VARCHAR2(50)	País do destino.
shipping_value	NUMBER(12,2)	Valor cobrado de frete.
load_datetime	TIMESTAMP DEFAULT SYSTIMESTAMP	Data/hora da inserção na staging. Usado para auditoria.

source_system	VARCHAR2(30) DEFAULT 'MONGODB'	Origem do dado (padrão: MongoDB). Permite expansão futura (ex.: API, Kafka).

2. Índices da Tabela STG_SALES

IDX_STG_SALES_ORDER_PROD

Propriedade	Valor
Tabela	STAGING.STG_SALES
Colunas	(order_id, product_id)
Finalidade	Acelera a identificação de itens por pedido e otimiza *joins* durante cargas dimensionais.

IDX_STG_SALES_IDMONGO

Propriedade	Valor
Tabela	STAGING.STG_SALES
Colunas	(id_mongo)
Finalidade	Garante velocidade na detecção de duplicidade dos documentos do MongoDB.

IDX_STG_SALES_ORDERDATE

Propriedade	Valor
Tabela	STAGING.STG_SALES
Colunas	(order_date_utc)
Finalidade	Otimiza cargas incrementais por data e facilita particionamento futuro.

Tabela: STAGING.ETL_EXECUTION_LOG

Descrição: tabela de auditoria utilizada para registrar detalhes de cada execução do pipeline ETL.

Coluna	Tipo	Descrição
execution_id	NUMBER (PK)	Identificador único da execução do pipeline. Pode ser preenchido via sequence.

start_time	TIMESTAMP DEFAULT SYSTIMESTAMP	Data/hora em que a execução iniciou.
end_time	TIMESTAMP	Data/hora de finalização da execução.
status	VARCHAR2(20)	Indicador da execução: SUCCESS, ERROR, RUNNING.
total_records	NUMBER	Quantidade de registros carregados na STAGING.
error_message	CLOB	Texto completo do erro quando o status é ERROR.

Anexo III - DW - Dicionário de Dados

Tabela: DIM_CUSTOMER

Coluna	Tipo	Regra / Observações
sk_customer	NUMBER (PK)	Surrogate Key gerada pela sequence + trigger. Identificador interno da dimensão.
bk_customer_id	VARCHAR2(30)	Deve ser único. Chave de negócio proveniente da origem (ex.: MongoDB).
name	VARCHAR2(200)	Nome do cliente. Alterações geram nova versão (SCD2).
email	VARCHAR2(200)	Email. Pode gerar nova versão (SCD2).
document	VARCHAR2(20)	CPF/CNPJ sem formatação.
dt_inicial	DATE	Data de início da validade do registro (controle SCD2).
dt_final	DATE	Data de final da validade. Versão ativa costuma ter 31/12/9999 ou NULL.
fl_ativo	CHAR(1)	'S' = ativo; 'N' = histórico.

Tabela: DIM_PRODUCT

Coluna	Tipo	Regra / Observações
sk_product	NUMBER (PK)	Surrogate Key gerada pela sequence + trigger. Identificador interno da dimensão.
bk_product_id	VARCHAR2(30)	Chave de negócio do produto. Deve ser única.
name	VARCHAR2(200)	Nome do produto. Alterações geram nova versão (SCD2).
category	VARCHAR2(100)	Categoria do produto (ex.: Eletrônicos). Alterações geram nova versão (SCD2).
dt_inicial	DATE	Data de início da validade do registro.
dt_final	DATE	Data de final de validade da versão. Para linha ativa: `31/12/9999` ou NULL.
fl_ativo	CHAR(1)	Flag indicando versão ativa (`'S'`) ou histórica (`'N'`).

Tabela: DIM_DATE

Coluna	Tipo	Regra / Observações
sk_date	NUMBER (PK)	Chave substituta. Geralmente no formato **AAAAMMDD**. Deve ser carregada pelo ETL. Não é auto-incremento.
date_value	DATE	Data completa do calendário.
year	NUMBER(4)	Ano correspondente.
month	NUMBER(2)	Mês numérico (1–12).
day	NUMBER(2)	Dia do mês (1–31).

Observações Importantes:

- A DIM_DATE é uma dimensão estática, muitas vezes pré-populada para um período de 10–20 anos.
- O sk_date normalmente segue o padrão AAAAMMDD, o que facilita joins e filtragem.
- Não utiliza trigger ou sequence.

Tabela: FCT_SALES

Coluna	Tipo	Regra / Observações
sk_sale	NUMBER (PK)	Surrogate Key gerada pela sequence + trigger. Identificador interno da dimensão.
sk_customer	NUMBER NOT NULL	FK para `DIM_CUSTOMER.sk_customer`. Obrigatório.
sk_product	NUMBER NOT NULL	FK para `DIM_PRODUCT.sk_product`. Obrigatório.
sk_order_date	NUMBER NOT NULL	FK para `DIM_DATE.sk_date`. Deve corresponder ao formato AAAAMMDD.
order_id	VARCHAR2(30)	Código do pedido na origem. Pode ser nulo para fatos agregados.
quantity	NUMBER(10)	Quantidade total do item vendido.
unit_price	NUMBER(12,2)	Valor unitário do item no momento da venda.
discount_value	NUMBER(12,2)	Valor do desconto aplicado.
shipping_value	NUMBER(12,2)	Valor do frete.

gross_amount	NUMBER(12,2)	Valor bruto = quantity × unit_price.
net_amount	NUMBER(12,2)	Valor líquido = gross_amount – discount + shipping.
payment_method	VARCHAR2(30)	Forma de pagamento (PIX, cartão, boleto...).
payment_status	VARCHAR2(30)	Status (pago, pendente, recusado...).
load_datetime	TIMESTAMP	Data/hora de inserção na fato. Default: `SYSTIMESTAMP`.

Observações Técnicas:

- Essa tabela faz parte do modelo estrela (Star Schema).
- É a fato de nível mais granular, uma linha por item de pedido.
- O cálculo de métricas (gross_amount, net_amount) normalmente é feito pelo pacote ETL.

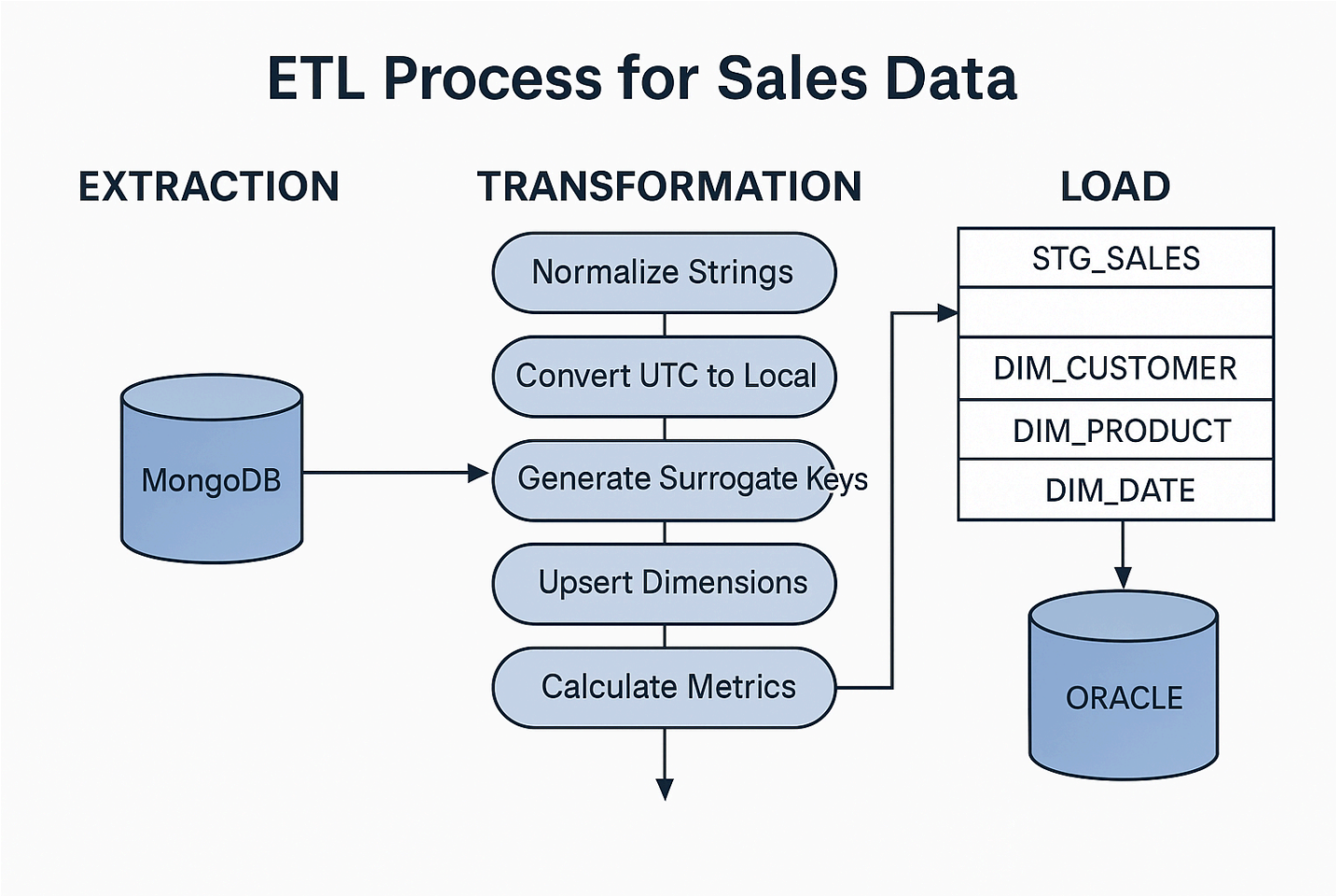
Anexo IV - Tabela Resumo – Regras do Enriquecimento (PKG_ETL_SALES)

Etapa	Regra / Comportamento
1. Normalização de Strings	Remover acentos; aplicar TRIM; converter para UPPER; remover caracteres especiais; padronizar atributos antes de comparar ou inserir.
2. Conversão de Datas (UTC → Brasil)	Converter `order_date_utc` para timezone `America/Sao_Paulo`; ajustar horário de verão automaticamente; gerar data final usada para SK_DATE.
3. Cálculo de `gross_amount`	`gross_amount = quantity * unit_price`.
4. Cálculo de `net_amount`	`net_amount = gross_amount - discount_value + shipping_value`.
5. Cálculo de `discount_percent`	`(discount_value / gross_amount) * 100`, quando houver desconto.
6. SCD Type 2 – Cliente	Verificar atributos modificados; encerrar versão atual (`dt_final`, `fl_ativo='N'`); inserir nova versão com `dt_inicial` atual; retornar novo `sk_customer`.
7. SCD Type 2 – Produto	Mesma regra do cliente: comparar atributos; encerrar versão antiga; inserir nova; manter histórico completo.
8. Chave de Data (SK_DATE)	Gerar surrogate key no formato AAAAMMDD; buscar ou inserir na DIM_DATE; retornar a chave.
9. Preparação para Fato	Garantir que todas as SKs (cliente, produto, data) estejam válidas antes da carga.
10. Inserção na FCT_SALES	Inserir linha com SKs e métricas calculadas; `load_datetime` preenchido automaticamente; manter integridade referencial.
11. Controle de Lote	Processar somente registros da STAGING com `batch_id` informado; cada lote gera sua própria carga.

Anexo V - DW - Triggers

Trigger	Tabela	Objetivo	Regra / Comportamento
TRG_DIM_CUSTOMER_PK	`DIM_CUSTOMER`	Gerar automaticamente a Surrogate Key (`sk_customer`)	Antes do INSERT: se `sk_customer` for NULL, atribuir `SEQ_DIM_CUSTOMER.NEXTVAL`.
TRG_DIM_PRODUCT_PK	`DIM_PRODUCT`	Gerar automaticamente a Surrogate Key (`sk_product`)	Antes do INSERT: se `sk_product` for NULL, atribuir `SEQ_DIM_PRODUCT.NEXTVAL`.
TRG_FCT_SALES_PK *(se utilizada)*	`FCT_SALES`	Gerar automaticamente a chave primária `sk_sale`	Antes do INSERT: se `sk_sale` for NULL, atribuir `SEQ_FCT_SALES.NEXTVAL`. *(opcional, dependendo do projeto)*
TRG_AUDIT_STG_SALES *(opcional)*	`STG_SALES`	Manter auditoria de carga	Antes do INSERT: preencher `load_datetime` com `SYSTIMESTAMP` caso não informado.
TRG_AUDIT_FACT *(opcional)*	`FCT_SALES`	Auditoria de operações na Fato	Após INSERT: gravar informações de auditoria em tabela de log.

Anexo VI - Processo de ETL



Anexo IV - Estrutura de Projeto Sugerido

```
etl_sales_pipeline/
├─ README.md
├─ requirements.txt
├─ config/
│  └─ config_example.yaml
├─ sql/
│  ├─ 01_staging_objects.sql
│  ├─ 02_dw_objects.sql
│  └─ 03_pkg_etl_sales.sql
└─ python/
   ├─ config_loader.py
   ├─ db.py
   ├─ mongo_extractor.py
   ├─ staging_loader.py
   ├─ etl_runner.py
   └─ main.py
```