**1. Arquitetura do Sistema**

**1.1) Camada persistência de Dados:**

**PostgreSQL:** Utilizado como Data Warehouse para armazenar dados históricos e estruturados. Implementado com modelo estrela para possibilitar análises OLAP, agregações por tempo, localidade, cliente e produto.

**MongoDB:** usado para armazenar dados semiestruturados e não estruturados, como feedbacks de clientes, avaliações e registros de vendas documentais.

**ObjectDB:** uso foi conceitual, representando a modelagem baseada em classes com relações diretas. Serviu como base para compreender a estrutura dos dados e apoiar a integração entre os modelos relacional e documental.

**1.2) Camada de integração (ETL e Consultas):**

Implementação do ETL:

**Extração:** Dados coletados de fontes simuladas por meio de scripts (inserir\_vendas.py, inserir\_comentarios.py), representando vendas e comentários de clientes.

**Transformação:** Antes da inserção, os dados são formatados em estruturas consistentes (Cálculo de valor\_total da venda [quantidade × preço], Normalização de nomes de produtos/clientes e etc)

**Carregar:** Os dados transformados são inseridos diretamente nas coleções MongoDB (vendasDB) via pymongo.

No PostgreSQL, os dados foram organizados segundo o modelo estrela para análise OLAP.

**2. Modelagem de Dados:**

**2.1) Data Warehouse (PostgreSQL)**

Modelagem esquema estrela:

**Tabela Fato** *fato\_vendas*: armazena as medidas quantitativas de negócio, como o valor\_total, e as chaves estrangeiras ligando às dimensões.

**Tabelas Dimensão**

*dim\_cliente*: informações de clientes (nome, faixa etária, sexo)

*dim\_produto*: informações dos produtos (nome, categoria)

*dim\_localidade*: localidade das vendas (cidade, estado)

*dim\_tempo*: data da venda (dia, mês, ano)

**2.2) MongoDB (PostgreSQL)**

Modelagem baseada em documentos JSON:

**coleção vendas:** cliente, produto, quantidade, valor\_total, data

**coleção comentários:** cliente\_id, produto, avaliacao, comentario, data

**2.3) ObjectDB (MODELAGEM CONCEITUAL)**

Modelagem orientada a objetos com classes representando: Cliente, Produto, Venda, Comentário. Não foi implementado tecnicamente, mas a modelagem conceitual serviu de base para as outras camadas.

**3. Implementação:**

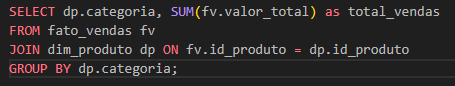
**3.1) Conexões com os bancos**

**PostgreSQL:** Uso das bibliotecas psycopg2 e SQLAlchemy para conexão, execução de queries e manipulação do DW.

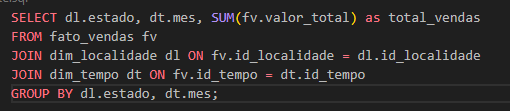
**MongoDB:** Conexão realizada por pymongo, com acesso à base vendasDB e coleções vendas e comentários.

**3.2) Consultas SQL (exemplo)**

**Total de vendas por categoria de produto:**



**Vendas por estado e mês:**

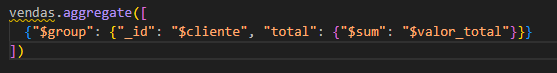


**3.2) Consultas MongoDB (exemplo)**

**Comentários positivos:**



**Total vendas por cliente:**



**3.3) Execução dos Scripts**

**consultas\_dw.py:** Executa consultas no Data Warehouse.

**consultas\_mongodb.py:** Acessa dados no MongoDB e imprime resultados.

**inserir\_vendas.py:** Popula a coleção vendas com dados de teste.

**inserir\_comentarios.py:** Popula a coleção comentários com dados de teste.

**4. Testes Realizados:**

**4.1) Testes no PostgreSQL**

Validação das consultas SQL

Checagem das junções entre dimensões e tabela fato

**4.2) Testes no MongoDB**

Inserção de dados teste

Filtros por data, avaliação, produto

**4.3) Integração Python**

Teste de conexões

Impressão formatada dos resultados das consultas