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:

```sql
SELECT dp.categoria, SUM(fv.valor_total) as total_vendas
FROM fato_vendas fv
JOIN dim_produto dp ON fv.id_produto = dp.id_produto
GROUP BY dp.categoria;
```

Vendas por estado e mês:

```sql
SELECT dl.estado, dt.mes, SUM(fv.valor_total) as total_vendas
FROM fato_vendas fv
JOIN dim_localidade dl ON fv.id_localidade = dl.id_localidade
JOIN dim_tempo dt ON fv.id_tempo = dt.id_tempo
GROUP BY dl.estado, dt.mes;
```

3.2) Consultas MongoDB (exemplo)
Comentários positivos:

```
comentarios.find({"avaliacao": {"$gte": 4}})
```

Total vendas por cliente:

```
vendas.aggregate([
  {"$group": {"_id": "$cliente", "total": {"$sum": "$valor_total"}}}
])
```

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