

Documentação do Código

Autora: Izabelle Cristinne da Silva Cruz

1. Descrição Geral

Este projeto é o teste para vaga de estágio da empresa Triggo.AI e utiliza Python, Pandas e SQL para realizar a limpeza, transformação e análise de dados a partir de um arquivo CSV contendo informações sobre clientes, vendedores, tipos de venda e valores. O projeto abrange desde a criação do banco de dados, tabelas e relacionamentos até o processamento e visualização das informações de vendas trimestrais. A integração com o banco de dados é feita utilizando a linguagem SQL e Python, enquanto o processamento de dados e a criação de gráficos são feitos com Python e as bibliotecas pandas, SQLAlchemy e Matplotlib.

2. Requisitos

Para rodar este projeto corretamente, você precisa de um ambiente de desenvolvimento configurado com as seguintes ferramentas e bibliotecas:


pandas: Para manipulação e processamento de dados.

SQLAlchemy: Para facilitar a interação com o banco de dados MySQL.

matplotlib: Para geração de gráficos de visualização dos resultados.

mysql-connector-python ou pymysql: Para conectar-se ao banco de dados MySQL.

Você pode instalar todas as dependências necessárias com os seguintes comandos no terminal da sua IDE.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It displays three installation commands, each preceded by a header in a different color: red for pandas, yellow for SQLAlchemy, and blue for matplotlib. The commands are in white text.

```
Instalar o pandas:  
pip install pandas  
  
Instalar o SQLAlchemy:  
pip install SQLAlchemy  
  
Instalar o matplotlib:  
pip install matplotlib
```

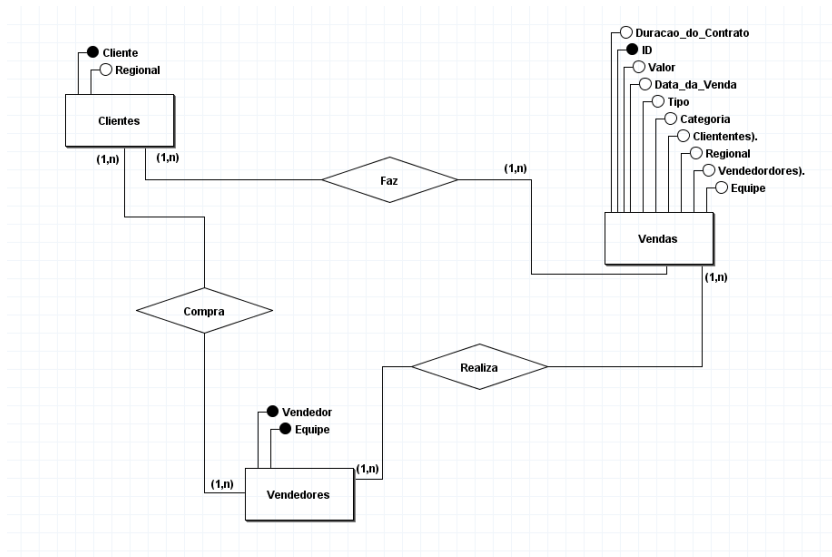
Sistema Operacional: O código pode ser executado em qualquer sistema operacional (Windows, macOS, Linux) que tenha as ferramentas e bibliotecas necessárias instaladas.

Banco de Dados Relacional: O projeto foi desenvolvido utilizando MySQL, mas pode ser adaptado para outros bancos como PostgreSQL, SQL Server, ou SQLite com pequenas modificações nas consultas e configurações.

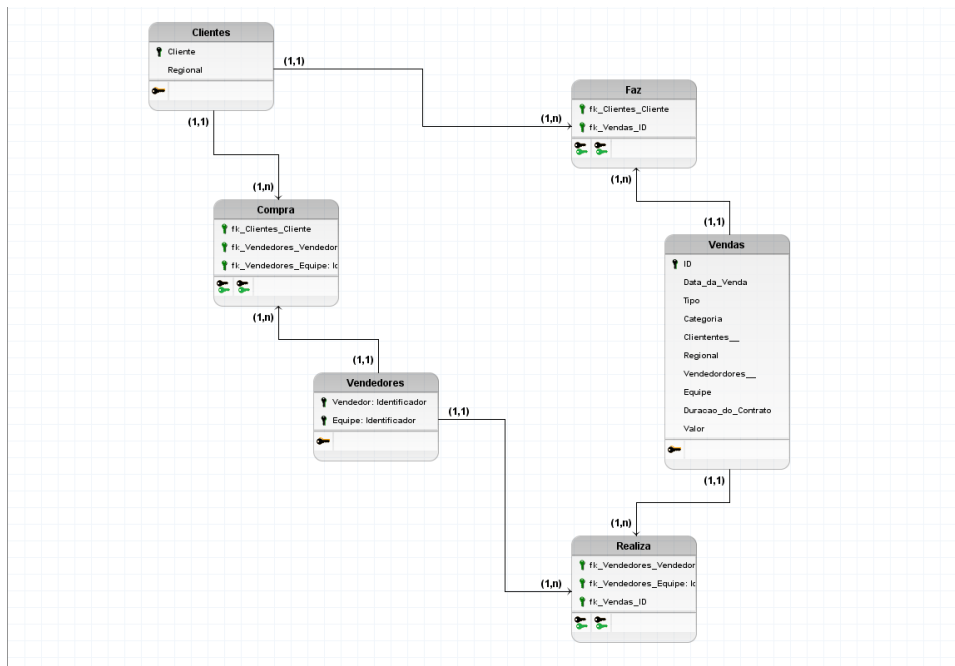
3. Modelo Lógico e Conceitual do Banco de Dados

Para executar a criação completa do banco de dados, foi necessário primeiramente modelá-lo, seguindo as etapas do modelo lógico e conceitual feito no brModelo.

- Modelo Conceitual do banco de dados



- Modelo Lógico do banco de dados



- Estrutura do Banco de Dados

O banco de dados foi estruturado com base em três tabelas principais: **Clientes**, **Vendedores** e **Vendas**.

Detalhes das Tabelas

1. **Clientes**
Contém informações sobre os clientes, identificados pela chave composta (*Cliente*, *Regional*). Essa estrutura permite identificar os clientes por região e facilita análises regionais.
2. **Vendedores**
Representa os vendedores e suas respectivas equipes, também utilizando uma chave composta (*Vendedor*, *Equipe*), garantindo que cada vendedor esteja associado a uma única equipe.
3. **Vendas**
Registra informações das vendas, como:
 - Data da venda
 - Tipo e categoria da venda
 - Identificadores do cliente, vendedor e equipe
 - Duração do contrato e valor

Relacionamentos

Os relacionamentos entre as tabelas foram modelados de forma a refletir os vínculos do sistema:

- Um cliente pode realizar múltiplas compras, mas cada compra está vinculada a uma única venda (relação 1:N).
- Um vendedor pode estar associado a várias vendas por meio da tabela intermediária **Realiza**.
- A tabela **Faz** foi projetada para representar a associação entre clientes e vendas.

Raciocínio por Trás da Modelagem

Ao projetar o banco de dados, levei os seguintes aspectos consideração:

1. Performance:

A definição de chaves primárias e estrangeiras foi feita para melhorar a performance das consultas e garantir consistência nos relacionamentos.

2. Escalabilidade:

A estrutura permite escalabilidade, sendo capaz de suportar o aumento no volume de dados sem comprometer a organização ou performance.

4. Código SQL

Primeiramente, você deve ter o MySQL instalado e rodando em sua máquina ou utilizar um servidor de banco de dados remoto. Após instalar o MySQL, execute o script SQL abaixo para criar o banco de dados e as tabelas necessárias para o projeto.

```
CREATE DATABASE IF NOT EXISTS ProjetoTriggoAI;
USE ProjetoTriggoAI;

CREATE TABLE IF NOT EXISTS Clientes (
  Cliente VARCHAR(255) NOT NULL,
  Regional VARCHAR(255) NOT NULL,
  PRIMARY KEY (Cliente, Regional)
);

CREATE TABLE IF NOT EXISTS Vendedores (
  Vendedor VARCHAR(255) NOT NULL,
  Equipe VARCHAR(255) NOT NULL,
  PRIMARY KEY (Vendedor, Equipe)
);

CREATE TABLE IF NOT EXISTS Vendas (
  ID BIGINT NOT NULL,
  Data_da_Venda DATE NOT NULL,
  Tipo VARCHAR(255) NOT NULL,
  Categoria VARCHAR(255) NOT NULL,
  Cliente VARCHAR(255) NOT NULL,
  Regional VARCHAR(255) NOT NULL,
  Vendedor VARCHAR(255) NOT NULL,
  Equipe VARCHAR(255) NOT NULL,
  Duracao_do_Contrato INT NOT NULL,
  Valor DECIMAL(18, 2) NOT NULL,
  PRIMARY KEY (ID),
  FOREIGN KEY (Cliente, Regional) REFERENCES Clientes(Cliente, Regional),
  FOREIGN KEY (Vendedor, Equipe) REFERENCES Vendedores(Vendedor, Equipe)
);

CREATE TABLE IF NOT EXISTS Vendas_Trimestrais (
  Ano INT NOT NULL,
  Trimestre INT NOT NULL,
  Total_Vendas DECIMAL(18, 2) NOT NULL,
  PRIMARY KEY (Ano, Trimestre)
);
```

Eu preferi começar por criar o código SQL antes mesmo de criar os Scripts em Python, pois assim eu já deixaria uma base das coisas que eu iria precisar fazer. Comecei criando

o banco de dados” ProjetoTriggoAI”, onde optei por usar o comando **IF NOT EXISTS** para evitar erros caso o banco já tenha sido criado anteriormente. Dei início ao uso do banco utilizando **USE** para garantir que todos os comandos subsequentes sejam executados no contexto desse banco de dados. A tabela **Clientes** foi criada para armazenar informações sobre os clientes e suas regiões de operação. Essa estrutura foi pensada para organizar dados que são essenciais para identificar de forma única um cliente em um determinado local.

Eu decidi juntar duas colunas para fazer uma PK para que não existam registros duplicados para o mesmo cliente em uma mesma região, pois como só existia a coluna ID no arquivo, eu não tinha como identificar os usuários de forma única sem que o código desse errado.

A tabela **Vendedores** foi projetada para registrar os vendedores e suas respectivas equipes. A relação entre vendedores e equipes é importante para análises, como o desempenho por equipe. A combinação de **Vendedor** e **Equipe** como chave primária assegura que um mesmo vendedor não esteja associado a mais de uma equipe dentro do sistema.

A tabela **Vendas** virou o centro do sistema, armazenando todas as transações realizadas. Decidi organizar os campos dessa forma para refletir todas as informações essenciais de cada venda.

- **ID:** Chave primária única para identificar cada venda.
- **Data_da_Venda:** Importante para rastrear as vendas realizadas ao longo do tempo e permitir análises temporais.
- **Tipo e Categoria:** Classificam as vendas, facilitando relatórios detalhados, como vendas por tipo de contrato ou categoria de serviço.
- **Cliente e Regional:** Relacionados diretamente à tabela Clientes, por meio de uma chave estrangeira, garantindo consistência.
- **Vendedor e Equipe:** Associados à tabela Vendedores, possibilitando rastrear o desempenho individual e por equipe.
- **Duracao_do_Contrato:** Um campo para armazenar a duração do contrato em meses.
- **Valor:** Utilizei o tipo DECIMAL(18, 2) para armazenar valores financeiros com precisão, considerando tanto grandes números quanto casas decimais.

E por último a tabela Vendas_Trimestrais, Essa tabela eu criei para armazenar os totais consolidados de vendas por trimestre. Ela me ajuda facilitando as análises de desempenho ao longo dos anos e permite gerar relatórios gerenciais mais rapidamente.

5. Código Python

Decidi modularizar o projeto em vários arquivos para garantir clareza, organização e principalmente facilidade na hora da manutenção. Por se tratar de um projeto de processamento de dados que normalmente já possui muitas etapas com banco de dados e geração de gráficos, eu escolhi trabalhar dessa forma. Cada arquivo foca em uma responsabilidade específica, tornando o código mais organizado e fácil de entender. Além de claro dividir o projeto em módulos permite reutilizar partes do código em outros projetos, como a classe de limpeza de dados (DataCleaner) ou funções auxiliares no arquivo helpers.py.

6. Explicação de Cada Arquivo

➤ Arquivo: **cleaner.py**

- Este arquivo contém a classe **DataCleaner**, que agrupa funções estáticas para padronização de colunas e limpeza de valores financeiros. Decidi isolar essa classe porque a limpeza de dados é uma etapa comum em vários projetos, e isso facilita na hora da reutilização.

➤ Arquivo: **helpers.py**

- Nesse arquivo estão funções auxiliares que me ajudaram em tarefas genéricas, como configuração de localidade (configure_locale) e formatação de valores monetários (format_money)

➤ Arquivo: **main.py**

- O arquivo principal que integra todos os módulos e eu decidi centralizar as chamadas de cada módulo aqui, para que fique claro o fluxo geral do programa e para separar a lógica de execução das funções específicas.

➤ Arquivo: **processing.py**

- Contém funções específicas para cálculos e análises, como calcular_maiores_e_menores_vendas. Eu decidi separar essas funções para manter a lógica de análise isolada de outras partes do projeto, como a integração com o banco de dados.

➤ Arquivo: **integracao_banco.py**

- Este arquivo gerencia a integração com o banco de dados, desde a leitura do arquivo CSV até a inserção dos dados nas tabelas SQL. Como essa funcionalidade é independente do processamento ou geração de gráficos, decidi separá-la, facilitando a manutenção e testes.

➤ Arquivo: **grafico.py**

- Esse arquivo é foi exclusivamente criado para geração de gráficos, como o de vendas trimestrais e eu decidi mantê-lo separado para evitar que a lógica de visualização interfira no processamento ou na manipulação de dados.

7. Desafios Python

1. Construa uma tabela auxiliar que sumarie o valor vendido por cada vendedor, ordenando do maior para o menor. A tabela também é criada em CSV e colocada na pasta do diretório onde se encontra os arquivos Python.

```
=== Total Vendido por Vendedor (Ordenado) ===
```

vendedor	
Vendedor 3	R\$ 166.220.477,72
Vendedor 7	R\$ 57.799.181,73
Vendedor 10	R\$ 52.212.055,49
Vendedor 8	R\$ 47.274.510,01
Vendedor 1	R\$ 30.929.480,27
Vendedor 9	R\$ 26.374.558,54
Vendedor 13	R\$ 24.837.304,87
Vendedor 18	R\$ 22.662.296,25
Vendedor 29	R\$ 19.076.866,26
Vendedor 4	R\$ 17.346.823,03
Vendedor 14	R\$ 7.558.041,75
Vendedor 20	R\$ 7.520.906,99
Vendedor 19	R\$ 6.513.184,00
Vendedor 2	R\$ 5.673.760,88
Vendedor 16	R\$ 4.879.238,00
Vendedor 6	R\$ 3.548.886,69
Vendedor 28	R\$ 3.307.137,75
Vendedor 12	R\$ 3.019.669,14
Vendedor 22	R\$ 2.150.000,00
Vendedor 21	R\$ 1.943.240,00
Vendedor 23	R\$ 1.941.880,09
Vendedor 11	R\$ 1.221.786,61
Vendedor 17	R\$ 1.133.803,11
Vendedor 5	R\$ 913.900,00
Vendedor 24	R\$ 583.970,00
Vendedor 30	R\$ 515.446,83
Vendedor 27	R\$ 420.000,00
Vendedor 26	R\$ 384.000,00
Vendedor 15	R\$ 86.504,61
Vendedor 25	R\$ 40.532,00

2. Imprima e identifique qual foi o cliente responsável pela venda com maior valor e com menor valor.

```
=== Venda Mais Alta ===  
Cliente: Cliente 89, Vendedor: Vendedor 7, Valor: R$ 24.970.500,00  
  
=== Venda Mais Baixa ===  
Cliente: Cliente 120, Vendedor: Vendedor 17, Valor: R$ 1.250,00
```

3. Imprima o valor médio por Tipo de venda (Serviços, Licenciamento, Produtos).

```
=== Valor Médio por Tipo de Venda ===  
tipo  
Licenciamento    R$ 983.448,77  
Produtos          R$ 893.409,19  
Serviços          R$ 314.567,94  
Name: valor, dtype: object
```

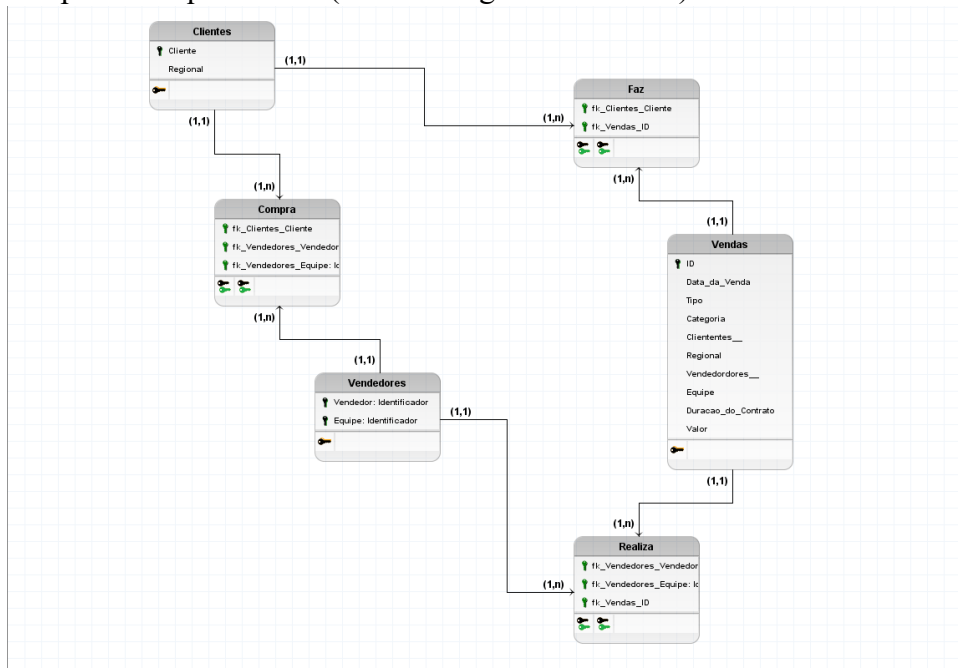
4. Imprima o número de vendas realizadas por cliente

```
=== Número de Vendas por Cliente ===  
Cliente 5: 128  
Cliente 8: 61  
Cliente 4: 60  
Cliente 10: 43  
Cliente 3: 31  
Cliente 13: 26  
Cliente 26: 25  
Cliente 2: 21  
Cliente 14: 21  
Cliente 30: 20  
Cliente 16: 19  
Cliente 52: 17  
Cliente 9: 17  
Cliente 31: 14  
Cliente 43: 13  
Cliente 24: 13  
Cliente 50: 11  
Cliente 12: 11  
Cliente 7: 10  
Cliente 44: 9  
Cliente 71: 9  
Cliente 37: 9  
Cliente 23: 8  
Cliente 34: 8  
Cliente 75: 8  
Cliente 76: 7  
Cliente 68: 7  
Cliente 21: 6  
Cliente 82: 6
```

```
=== A quantidade total de vendas é 853 ===
```

8. Desafios SQL

1. Construa o modelo de relacionamento com as categorias utilizadas em todos os campos do arquivo CSV (incluir imagem do modelo).



2.

3. Liste todas as vendas (ID) e seus respectivos clientes apenas no ano de 2020.

	id_vendas	Nome_Cliente	Data_da_Venda
▶	20202039	Cliente 10	2020-01-16
	20202113	Cliente 10	2020-02-18
	20202161	Cliente 10	2020-03-17
	20202162	Cliente 10	2020-03-30
	20202175	Cliente 10	2020-04-09
	20202233	Cliente 10	2020-04-01
	20202234	Cliente 10	2020-04-08
	20202337	Cliente 10	2020-06-30
	20202341	Cliente 10	2020-05-19
	20202398	Cliente 10	2020-06-09
	20202399	Cliente 10	2020-06-19
	20202401	Cliente 10	2020-06-09
	20202471	Cliente 10	2020-07-07
	20202583	Cliente 10	2020-09-28
	20202594	Cliente 10	2020-10-19
	20202605	Cliente 10	2020-08-24
	20202206	Cliente 100	2020-05-15
	20191169	Cliente 101	2020-05-26
	20202319	Cliente 101	2020-05-26
	20202693	Cliente 102	2020-06-04
	20202394	Cliente 103	2020-07-07
	20203105	Cliente 104	2020-08-01
	20202530	Cliente 105	2020-08-03
	20202567	Cliente 106	2020-08-17
	20202390	Cliente 107	2020-08-17
	20202418	Cliente 108	2020-08-18
	20203179	Cliente 108	2020-08-18

4. Liste a equipe de cada vendedor.

	Vendedor	Equipe
▶	Vendedor 1	Time 1
	Vendedor 10	Time 3
	Vendedor 10	Time 7
	Vendedor 11	Time 3
	Vendedor 12	Time 10
	Vendedor 12	Time 6
	Vendedor 13	Time 3
	Vendedor 14	Time 3
	Vendedor 15	Time 3
	Vendedor 16	Time 3
	Vendedor 17	Time 8
	Vendedor 18	Time 1
	Vendedor 19	Time 3
	Vendedor 2	Time 2
	Vendedor 20	Time 3
	Vendedor 21	Time 3
	Vendedor 22	Time 11
	Vendedor 23	Time 1
	Vendedor 24	Time 11
	Vendedor 25	Time 3
	Vendedor 26	Time 12
	Vendedor 27	Time 12

5. Construa uma tabela que avalie trimestralmente o resultado de vendas e plote um gráfico deste histórico

	Ano	Trimestre	Total_Vendas_Formatado
▶	2018	1	9.214.708,51
	2018	2	24.329.411,79
	2018	3	17.807.820,99
	2018	4	28.684.705,77
	2019	1	14.924.051,07
	2019	2	48.639.438,83
	2019	3	35.105.020,45
	2019	4	41.333.096,43
	2020	1	72.307.629,80
	2020	2	49.069.698,10
	2020	3	52.073.231,25
	2020	4	43.522.269,02
	2021	1	47.535.693,63
6.	2021	2	33.542.666,98

