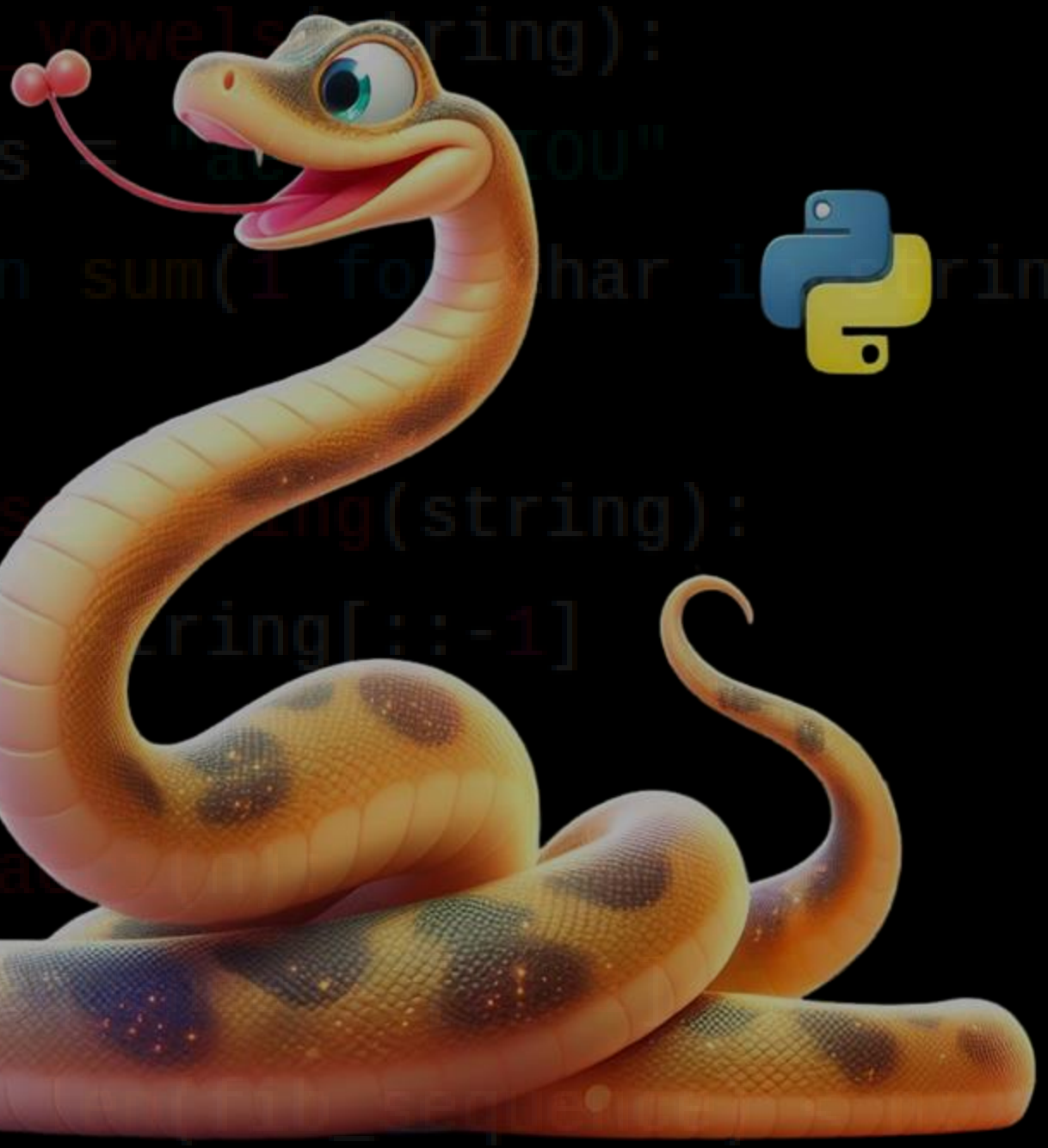
AUTOMAÇÕES COM PYTHON

Descomplicando a importação de dados



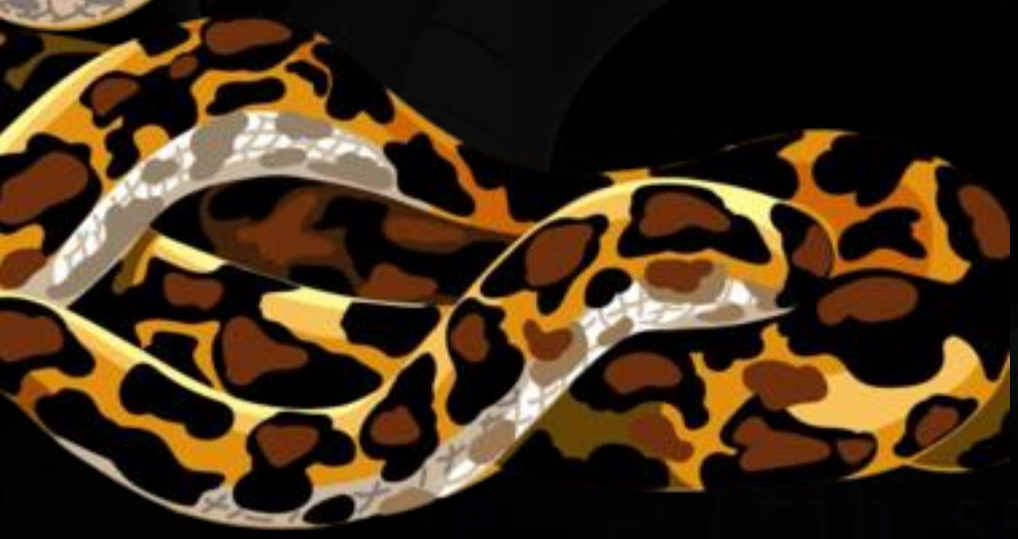
Edinaldo F. C. Santos

PROBLEMAS DO DIA A DIA

Imagine que no seu trabalho, diariamente, você receba uma base com os dados de vendas da sua empresa e que tenha que subir este arquivo para o banco de dados do SQL Server. Primeiro, você teria que abrir o arquivo, tratar os dados para ficar no formato que o importador do SQL entenda, salvar o arquivo em CSV/TXT (caso esteja em outro formato), abrir o importador do SQL e... ah, cansei só de falar. Mas chega de motivos para se preocupar! Vamos resolver isso com Python de forma rápida e simples. Vem comigo!

01

INSTALANDO AS BIBLIOTECAS



BIBLIOTECAS NECESSÁRIAS

Para facilitar nosso trabalho, vamos precisar de apenas três bibliotecas: pandas,sqlalchemy e pyodbc. Para instalá-lás, no editor de sua preferência (partindo do pressuposto que você já tenha o python instalado no seu computador), digite o seguinte comando:

```
pip install pandas sqlalchemy pyodbc
```

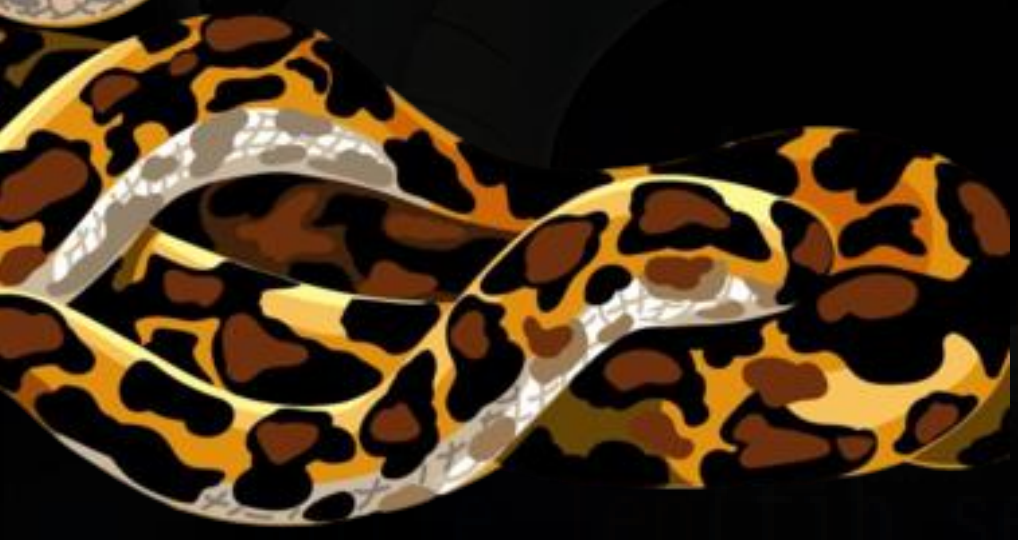
Pandas: Facilita a manipulação e análise de dados, como os de um CSV (dentre diversos outros formatos), através de estruturas de dados poderosas.

SQLAlchemy: Permite interagir com bancos de dados SQL de forma flexível e intuitiva, facilitando operações como consultas, inserções e atualizações de dados.

PyODBC: Fornece uma interface para conectar Python a bancos de dados através de ODBC (Open Database Connectivity)

02

ABRINDO O ARQUIVO



LENDO OS DADOS COM PANDAS

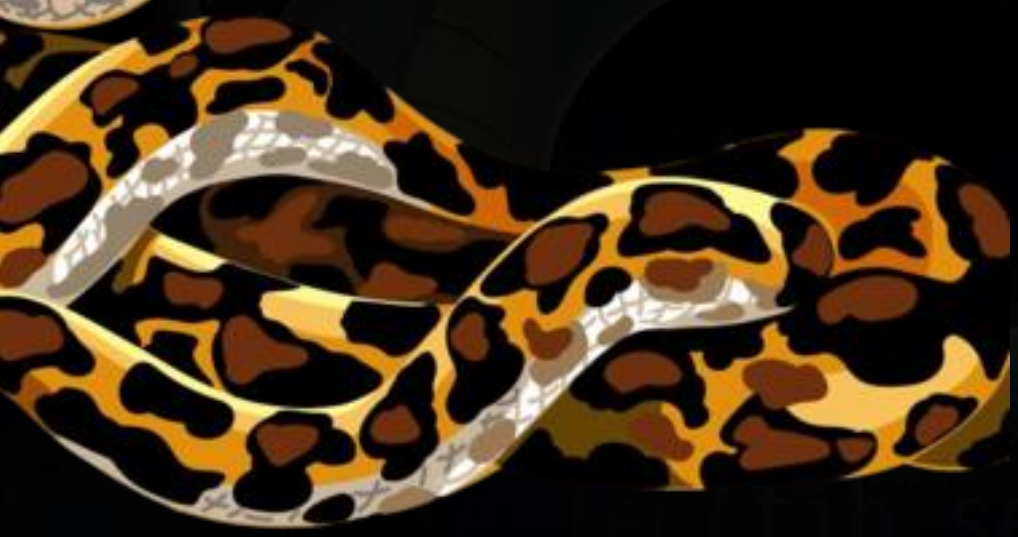
Agora, vamos abrir nosso arquivo de vendas usando o Pandas. Caso queira, pode usar o csv que está neste repositório, para fins de didáticos:
https://raw.githubusercontent.com/edinaldofcs/DIO_CHALLENGES/desenvolvimento/EBOOK/tabela_vendas.csv

```python
import pandas as pd

# Lembre de colocar o caminho do seu csv no lugar da url
base_url = "https://raw.githubusercontent.com/"
user = "edinaldofcs"
repo_branch = "/DIO_CHALLENGES/desenvolvimento/EBOOK/"
url = base_url + user + repo_branch + 'tabela_vendas.csv'

# Lendo o arquivo
df = pd.read_csv(url)
```

Pronto! Com apenas algumas linhas de código, já temos nossos dados no Pandas DataFrame. Simples e fácil!

03

FORMATANDO PARA O SQL SERVER



TRANSFORMANDO DADOS

Para garantir que nossos dados estão prontos para o SQL Server, precisamos fazer algumas modificações. Vamos garantir que as colunas e seus respectivos conteúdos estejam no formato correto.

O primeiro passo será ajustar os nomes das colunas, para ficarem iguais às que já existem no banco, ou seguindo boas práticas para o caso de uma nova tabela.

```python
# Remover espaços nos nomes das colunas
df.columns = df.columns.str.strip()

# Substituir espaços por underscores nos nomes das colunas
df.columns = df.columns.str.replace(' ', '_')

# Converter nomes das colunas para letras minúsculas
df.columns = df.columns.str.lower()
```

Definir os nomes adequadamente de acordo com os dados da coluna é de suma importância, pois, dependendo da situação, outra pessoa pode vir a usar esta tabela. Pense nisso!

TRANSFORMANDO DADOS

Com as colunas formatadas, vamos partir para a formatação dos tipos de dados das colunas de data, preco_unitario e valor_total.

Subir as colunas com o formato correto garante a integridade dos dados e a eficiência das consultas e operações realizadas no banco.

As datas, devem ser convertidas para o formato aaaa-mm-dd (ano-mes-dia), que é o formato aceito pelo SQL SERVER e pela maioria dos bancos de dados

```python
# Converter a coluna de datas para o formato do SQL Server
df['data'] = pd.to_datetime(df['data'], format='%d/%m/%Y')
```

TRANSFORMANDO DADOS

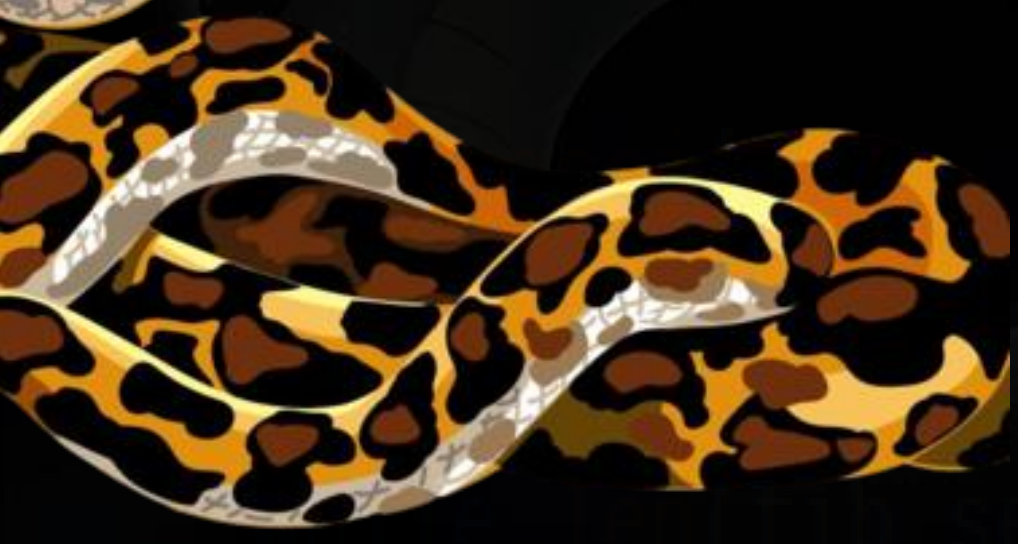
As colunas de preco_unitario e valor_total, devem ser convertidas para o tipo float, tendo em vista as casas decimais

```
# Converter as colunas de valores para formato numérico adequado (float)
df['preco_unitario'] = df['preco_unitario'].str.replace(',', '.').astype(float)
df['valor_total'] = df['valor_total'].str.replace(',', '.').astype(float)
```

Pronto! Agora já temos todas as colunas necessárias com a tipagem adequada e podemos passar para a próxima etapa

04

PROTEGENDO DADOS SENSÍVEIS



SEGURANÇA DAS INFORMAÇÕES

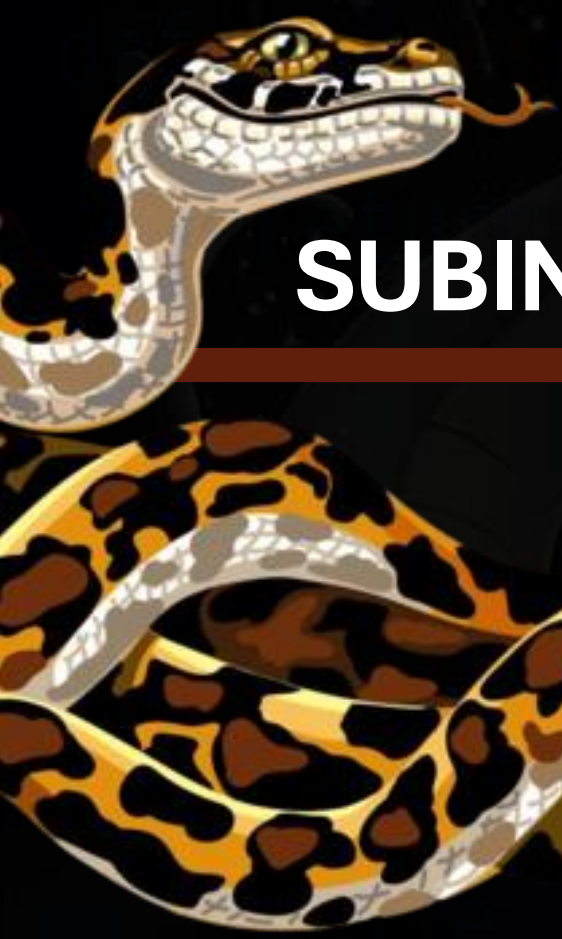
Para evitar que seus dados de conexão com o SQL SERVER fiquem expostos, é importante criarmos um arquivo separado com os dados necessário para a conexão com o banco. Para isso, vamos criar um arquivo chamado config.py com as informações necessárias.

```
server = 'NOME DO SERVIDOR'
database = 'NOME DO BANCO'
driver = 'ODBC Driver 17 for SQL Server'
```

Agora, podemos importar a biblioteca sqlalchemy (que será utilizada logo a seguir) e o arquivo config.py, logo abaixo da importação do pandas (VER CAPITULO 2)

```
import pandas as pd
from sqlalchemy import create_engine, Integer, String, Float, Date
from config import server,database,driver
```

05



SUBINDO DADOS PARA O BANCO

DEFININDO O DICIONARIO COM AS TIPAGENS

Antes de finalizarmos, vamos defini qual será o tipo dos dados da nossa tabela. Embora a tabela já exista no banco para o nosso exemplo, teremos situações onde criaremos tabelas novas e para isso devemos tipar as colunas do banco, conforme fizemos com as colunas do dataframe, caso contrário, tudo será considerado como VARCHAR para as colunas as quais os tipos ainda não foram identificados.

```python
from sqlalchemy import Integer, String, Float, Date

tipos_de_dados = {
    'id_venda': Integer(),
    'data': Date(),
    'cliente': String(100),
    'produto': String(100),
    'quantidade': Integer(),
    'preco_unitario': Float(),
    'valor_total': Float()
}
```

SUBINDO OS DADOS PARA O BANCO

Agora que já temos tudo pronto, podemos criar a conexão e subir os dados para o SQL SERVER.

```python
# Criar a string de conexão usando as informações do arquivo config.py
connection_string = f'mssql+pyodbc://{server}/{database}?driver={driver}'

# Criar o objeto de conexão com o banco de dados
engine = create_engine(connection_string)
# Nome da tabela
table_name = 'tabela_vendas'

print("Subindo para o Banco de dados...Aguarde")
df.to_sql(table_name, engine, index=False, if_exists='replace',
dtype = tipos_de_dados)

print("Processo finalizado")
```

Nos parâmetros da função "to_sql", utilizamos o parâmetro "append", pois, caso a tabela já exista, vamos apenas adicionar novos dados, mas também temos as opções "fail" e "replace". Qualquer dúvida, vale uma consulta na documentação que você pode encontrar no link abaixo:
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.to_sql.html

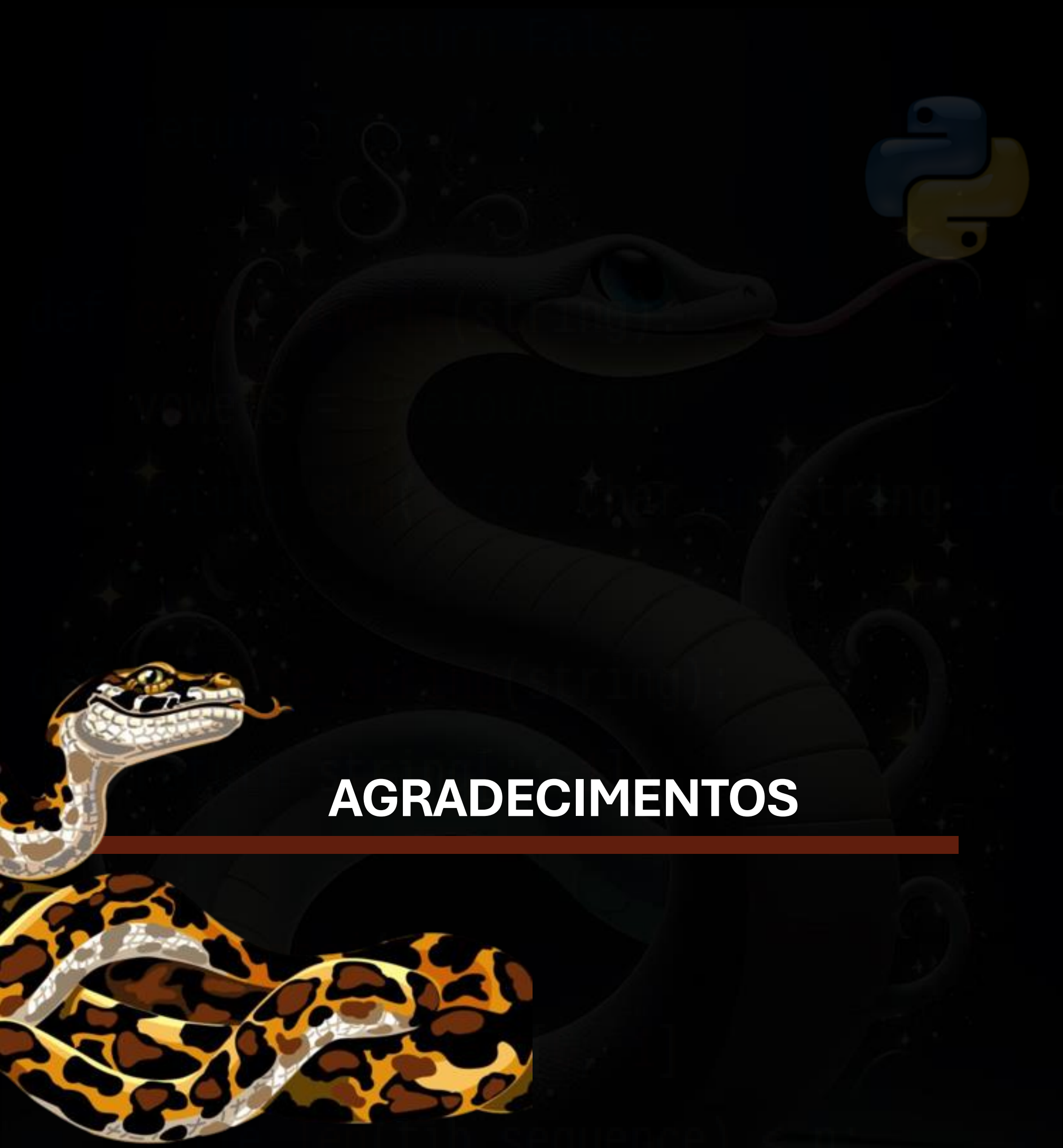
06

REFATORANDO

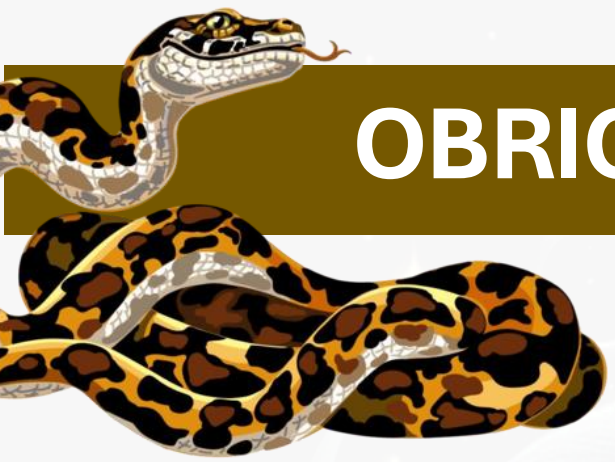
HORA DE REFATORAR

Por fim, podemos unir todas as partes em um código só, de maneira organizada e aplicando boas práticas

```python
import pandas as pd
from sqlalchemy import create_engine, Integer, String, Float, Date
from config import server,database,driver

def subir_para_o_banco_de_dados():
    try:
        base_url = "https://raw.githubusercontent.com/"
        user = "edinaldofcs"
        repo_branch = "/DIO_CHALLENGES/desenvolvimento/EBOOK/"
        url = base_url + user + repo_branch + 'tabela_vendas.csv'

        connection_string = f'mssql+pyodbc://{server}/{database}?driver={driver}'
        engine = create_engine(connection_string)

        df = pd.read_csv(url, dtype=str)
        df.columns = df.columns.str.strip()
        df.columns = df.columns.str.replace(' ', '_')
        df.columns = df.columns.str.lower()
        df['data'] = pd.to_datetime(df['data'], format='%d/%m/%Y').dt.date
        tipos_de_dados = {'id_venda': Integer(),'data': Date(),'cliente': String(100),
            'produto': String(100),'quantidade': Integer(),'preco_unitario': Float(),
            'valor_total': Float()
        }
        df['preco_unitario'] = df['preco_unitario'].str.replace(',', '.').astype(float)
        df['valor_total'] = df['valor_total'].str.replace(',', '.').astype(float)

        print("Subindo para o Banco de dados...Aguarde")
        table_name = 'tabela_vendas'
        df.to_sql(table_name, engine, index=False, if_exists='replace',
        dtype = tipos_de_dados)
        print("Processo finalizado")
    except Exception as e:
        print(f"Ocorreu um erro: {str(e)}")

subir_para_o_banco_de_dados()
```



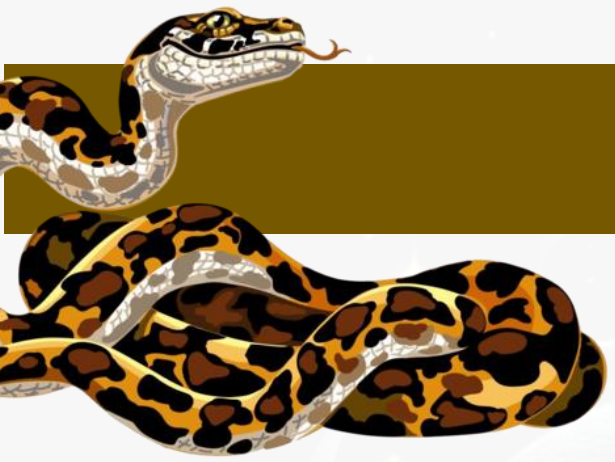
AGRADECIMENTOS



OBRIGADO PELO SEU TEMPO

Viu só, pessoal? Com algumas linhas de código em Python, conseguimos ler, transformar e subir dados para o SQL Server de maneira rápida e eficiente. Agora vocês estão prontos para enfrentar este tipo de desafio! 🚀

Embora, à primeira vista, possa parecer complicado, uma vez finalizado, este simples código permitirá repetibilidade. Ou seja, você pode executá-lo todos os dias ou sempre que precisar subir informações do mesmo tipo. Isso te poupará preciosos minutos do seu dia, que você pode aproveitar para automatizar outras tarefas. Também é possível já deixar o script agendado para rodar sempre em determinado horário, mas isso já é assunto para um outro post.



EM TEMPO...

Muito obrigado por chegar até aqui!

Boa parte deste ebook foi gerado por IA e diagramado por um humano (Eu). O passo a passo está no meu github.

https://github.com/edinaldofcs/DIO_CHALLENGES/tree/desenvolvimento/EBOOK