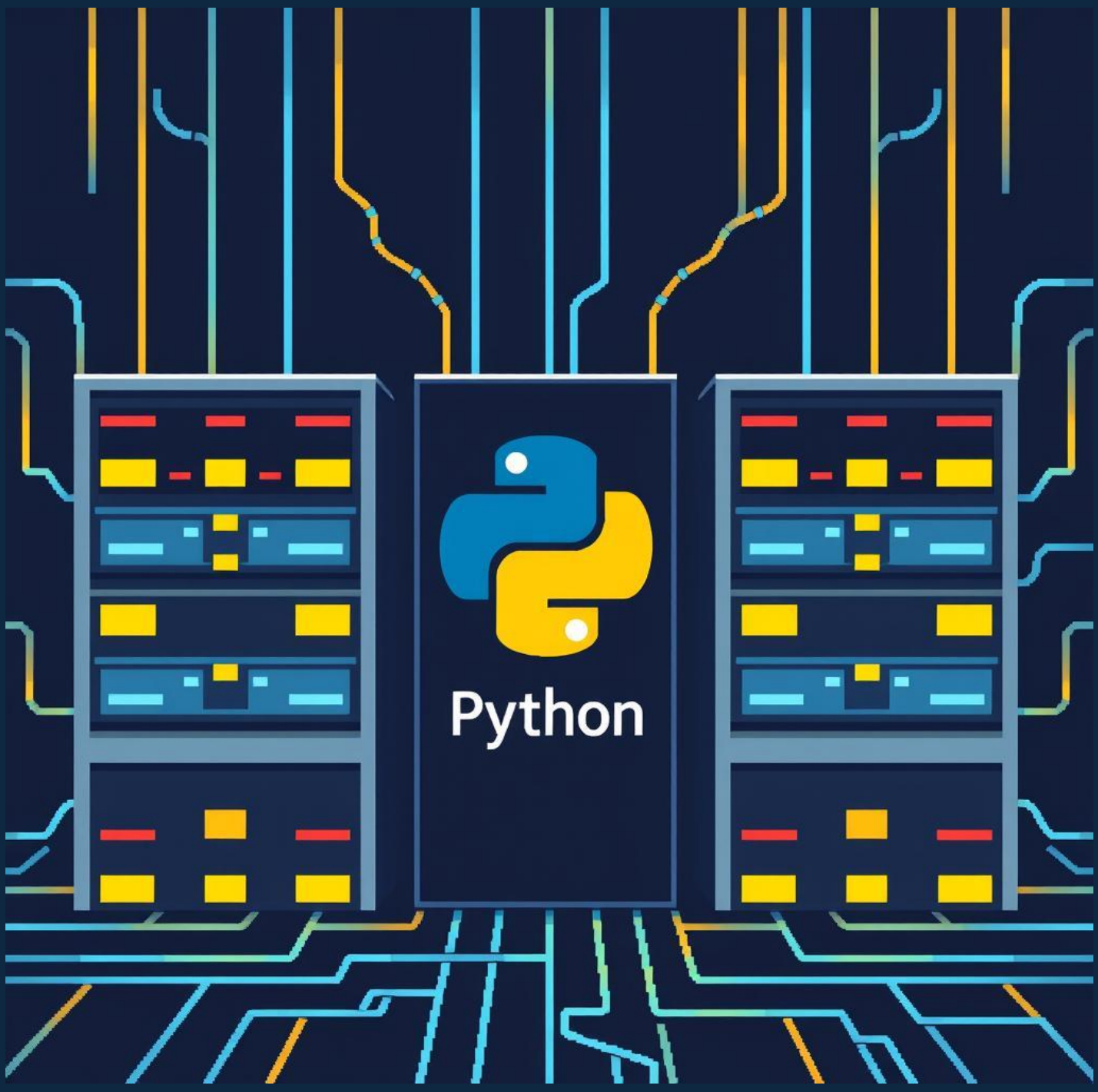
DataFrames em Fusão



O Poder do Pandas

Pandas

Pandas é a principal biblioteca de manipulação de dados em Python. Com ele, analisamos, transformamos e combinamos informações com eficiência. Neste ebook, vamos explorar o merge(), uma das ferramentas mais poderosas para cruzamento de dados em DataFrames.

Por que usar merge no Pandas?

Quando lidamos com dados reais, muitas vezes eles vêm separados em diferentes fontes ou tabelas. O `merge` do Pandas é a forma de unir essas informações com precisão, de forma parecida com os JOINs do SQL.

O básico do merge()

O método `merge( )` permite combinar dois DataFrames com base em colunas comuns.

Merge

```python
import pandas as pd

df_usuarios = pd.DataFrame({
    'id': [1, 2, 3],
    'nome': ['Ana', 'Bruno', 'Carlos']
})

df_compras = pd.DataFrame({
    'id': [1, 2, 2],
    'compra': ['Livro', 'Caderno', 'Lápis']
})

resultado = pd.merge(df_usuarios, df_compras, on='id')

print(resultado)
```

```
   id   nome   compra
0   1    Ana    Livro
1   2  Bruno  Caderno
2   2  Bruno    Lápis
```

Tipos de junção (how)

O parâmetro how define o tipo de junção: inner, left, right ou outer.

Merge

```python
import pandas as pd

df_usuarios = pd.DataFrame({
    'id': [1, 2, 3],
    'nome': ['Ana', 'Bruno', 'Carlos']
})

df_compras = pd.DataFrame({
    'id': [1, 2, 2],
    'compra': ['Livro', 'Caderno', 'Lápis']
})

resultado = pd.merge(df_usuarios, df_compras, on='id', how='outer')
print(resultado)
```

```
   id   nome   compra
0   1    Ana    Livro
1   2  Bruno  Caderno
2   2  Bruno    Lápis
3   3  Carlos     NaN
```

Junção com colunas diferentes

Quando as colunas têm nomes diferentes usamos o left_on e right_on:

Merge

```
import pandas as pd

df_clientes = pd.DataFrame({
    'cliente_id': [1, 2, 3],
    'nome': ['Ana', 'Bruno', 'Carlos']
})

df_pedidos = pd.DataFrame({
    'user_id': [2, 3],
    'pedido': ['Notebook', 'Mouse']
})

resultado = pd.merge(
    df_clientes,
    df_pedidos,
    left_on='cliente_id',
    right_on='user_id')
print(resultado)

   cliente_id   nome  user_id    pedido
0           2  Bruno        2  Notebook
1           3 Carlos        3     Mouse
```

Evitando conflitos com **suffixes**

Se os dois DataFrames tiverem colunas com o mesmo nome (além da chave), use suffixes:

Merge

```python
import pandas as pd

df1 = pd.DataFrame({
    'id': [1, 2],
    'nome': ['Ana', 'Bruno']
})

df2 = pd.DataFrame({
    'id': [1, 2],
    'nome': ['Notebook', 'Tablet']
})

resultado = pd.merge(df1, df2, on='id', suffixes=('_cliente', '_produto'))
print(resultado)
```

```
   id nome_cliente nome_produto
0   1          Ana     Notebook
1   2        Bruno       Tablet
```

Múltiplas chaves

Você pode fazer o merge com mais de uma coluna como chave:

Merge

```python
import pandas as pd

df_a = pd.DataFrame({
    'nome': ['Ana', 'Ana', 'Bruno'],
    'data': ['2024-01-01', '2024-01-02', '2024-01-01'],
    'venda': [100, 150, 200]
})

df_b = pd.DataFrame({
    'nome': ['Ana', 'Bruno'],
    'data': ['2024-01-02', '2024-01-01'],
    'regiao': ['Sul', 'Norte']
})

resultado = pd.merge(df_a, df_b, on=['nome', 'data'])
print(resultado)
```

```
   nome        data  venda regiao
0   Ana  2024-01-02    150    Sul
1 Bruno  2024-01-01    200  Norte
```

Conclusão prática

O merge() é essencial para unir dados com lógica e segurança. Com ele, é possível integrar tabelas complexas, cruzar múltiplas fontes e preparar dados com qualidade para análise.

Sempre pense no relacionamento entre os dados, escolha o tipo certo de junção e evite conflitos com boas práticas.

Merge bem feito é dado limpo!