desafio12

ME315-2S2025

brotto

2025-10-09 11:33:44.502043

Sumário

§ Natureza

Desafio 12 proposto na 20.ª aula de ME315-2S2025.

§ Python

Objetivos

- Introduzir o uso de SQLite com Polars
- Explorar cálculos estatísticos em bancos de dados SQLite3
- Realizar operações de agregação e filtragem usando SQL
- **Lembrete**: Os conhecimentos adquiridos com SQLite são diretamente aplicáveis a outros bancos de dados em SQL (MariaDB, MySQL, PostgreSQL).

```
## Inicia sessão python
   venv_python = "temp.venv"
   if(!virtualenv_exists(venv_python)){ virtualenv_create(venv_python, packages =
    ↪  c("pip", "setuptools")) }
```

```
## Using Python: C:/Users/ra260181/AppData/Local/r-reticulate/r-reticulate/pyenv/pyenv-
win/versions/3.12.10/python.exe
## Creating virtual environment "temp.venv" ...
## Done!
## Installing packages: pip, wheel, setuptools
## Virtual environment 'temp.venv' successfully created.
```

```
   py_install("polars", envname = venv_python)
```

```
## Using virtual environment "temp.venv" ...
```

```
   py_install("pyarrow", envname = venv_python)
```

```
## Using virtual environment "temp.venv" ...
```

```
   use_virtualenv(venv_python)
```

Requisitos

- Python instalado
- Biblioteca Polars instalada
- SQLite disponível no sistema

```
import polars as pl
import sqlite3
import os
```

Iniciando a conexão com o banco

- A conexão com o banco de dados é feita com o método **connect**.

- Obtemos também o **cursor**, que é um objeto que permite interagir com o banco de dados

```
# Lista todos os arquivos no diretório atual
arquivos = os.listdir('.')
print("Arquivos no diretório:", arquivos)
```

```
## Arquivos no diretório: ['.Rhistory', 'data.db', 'desafio12-me315.Rmd']
```

```
# Verifica especificamente pelo data.db
if 'data.db' in arquivos:
    os.remove('data.db')
    print("data.db removido!")
else:
    print("data.db não encontrado")
```

```
## data.db removido!
```

```
conn = sqlite3.connect("data.db")
cursor = conn.cursor()
```

Criação manual de uma tabela

- O método **execute** é empregado para executar comandos dentro do banco SQL.

```
cursor.execute('''
CREATE TABLE IF NOT EXISTS vendas (
    id INTEGER PRIMARY KEY,
    vendedor TEXT,
    produto TEXT,
    valor REAL,
    data_venda DATE
)
''')
```

```
## <sqlite3.Cursor object at 0x000001EF5ECA11C0>
```

Inserção de dados em uma tabela

- Ao realizar uma inserção, deve-se executar o **commit**, que fará a confirmação da operação.

```
cursor.execute('''
INSERT INTO vendas (vendedor, produto, valor, data_venda)
VALUES
    ('Ana', 'Produto A', 120.5, '2024-09-01'),
    ('Carlos', 'Produto B', 200.0, '2024-10-02'),
    ('Ana', 'Produto C', 150.0, '2024-09-03'),
    ('Bruno', 'Produto A', 300.0, '2024-11-04'),
    ('Carlos', 'Produto C', 100.0, '2024-10-05');
''')
```

```
## <sqlite3.Cursor object at 0x000001EF5ECA11C0>
```

```
conn.commit() # confirma uma operação
```

Consulta simples no SQLite

```
cursor.execute("SELECT * FROM vendas")
```

```
## <sqlite3.Cursor object at 0x000001EF5ECA11C0>
```

```
rows = cursor.fetchall()
for row in rows:
  print(row)
```

```
## (1, 'Ana', 'Produto A', 120.5, '2024-09-01')
## (2, 'Carlos', 'Produto B', 200.0, '2024-10-02')
## (3, 'Ana', 'Produto C', 150.0, '2024-09-03')
## (4, 'Bruno', 'Produto A', 300.0, '2024-11-04')
## (5, 'Carlos', 'Produto C', 100.0, '2024-10-05')
```

Integração com Polars

```
import polars as pl
dados = pl.read_database("SELECT * FROM vendas", conn)
print(dados)
```

```
## shape: (5, 5)
##
##   id    vendedor   produto     valor   data_venda
##   ---   ---        ---         ---     ---
##   i64   str        str         f64     str
##
##   1     Ana        Produto A   120.5   2024-09-01
##   2     Carlos     Produto B   200.0   2024-10-02
##   3     Ana        Produto C   150.0   2024-09-03
##   4     Bruno      Produto A   300.0   2024-11-04
##   5     Carlos     Produto C   100.0   2024-10-05
##
```

Exemplos

- Qual é o total de vendas por vendedor ?

```
vendas_total = pl.read_database('''
   SELECT vendedor, SUM(valor) as total_vendas
   FROM vendas
   GROUP BY vendedor;
''', conn)
print(vendas_total)
```

```
## shape: (3, 2)
##
##   vendedor   total_vendas
##   ---        ---
##   str        f64
##
##   Ana        270.5
##   Bruno      300.0
##   Carlos     300.0
```

##

- Qual é o valor médio de venda por vendedor ?

```
vendas_medias = pl.read_database('''
   SELECT vendedor, AVG(valor) as total_vendas
   FROM vendas
   GROUP BY vendedor;
''', conn)
print(vendas_medias)
```

```
## shape: (3, 2)
##
##   vendedor   total_vendas
##   ---        ---
##   str        f64
##
##   Ana        135.25
##   Bruno      300.0
##   Carlos     150.0
##
```

- Crie uma tabela contendo o nome do vendedor, o número de vendas realizadas, o total vendido e o valor médio por venda.

```
vendas_comb = pl.read_database("""
SELECT vendedor,
       COUNT(*) as numero_vendas,
       SUM(valor) as total_vendas,
       AVG(valor) as media_vendas
FROM vendas
GROUP BY vendedor;
""", conn)
```

Exemplo (continuação)

```
print(vendas_comb)
```

```
## shape: (3, 4)
##
##   vendedor   numero_vendas   total_vendas   media_vendas
##   ---        ---             ---            ---
##   str        i64             f64            f64
##
##   Ana        2               270.5          135.25
##   Bruno      1               300.0          300.0
##   Carlos     2               300.0          150.0
##
```

Exemplo

- Quais foram as vendas de, pelo menos, $200.00 ?

```
ticket_alto = pl.read_database("""
SELECT * FROM vendas WHERE valor >= 200
```

```
""", conn)
print(ticket_alto)
```

```
## shape: (2, 5)
##
##   id    vendedor   produto     valor   data_venda
##   ---   ---        ---         ---     ---
##   i64   str        str         f64     str
##
##   2     Carlos     Produto B   200.0   2024-10-02
##   4     Bruno      Produto A   300.0   2024-11-04
##
```

Operações com Datas em SQLite

- Qual foi o volume mensal de vendas ?

```
vendas_mensais = pl.read_database("""
SELECT strftime('%Y-%m', data_venda) AS mes, SUM(valor) AS total_vendas
FROM vendas GROUP BY mes ORDER BY mes
""", conn)
print(vendas_mensais)
```

```
## shape: (3, 2)
##
##   mes       total_vendas
##   ---       ---
##   str       f64
##
##   2024-09   270.5
##   2024-10   300.0
##   2024-11   300.0
##
```

Criando a tabela de produtos

```
cursor.execute('''
CREATE TABLE IF NOT EXISTS produtos (
    id INTEGER PRIMARY KEY,
    nome TEXT NOT NULL,
    categoria TEXT NOT NULL,
    preco REAL NOT NULL,
    estoque INTEGER NOT NULL
);
''')
```

```
## <sqlite3.Cursor object at 0x000001EF5ECA11C0>
```

```
cursor.execute('''
INSERT INTO produtos (nome, categoria, preco, estoque) VALUES
    ('Produto A', 'Categoria 1', 100.0, 50),
    ('Produto B', 'Categoria 2', 150.0, 30),
    ('Produto C', 'Categoria 1', 200.0, 20),
    ('Produto D', 'Categoria 2', 250.0, 10),
```

```
    ('Produto E', 'Categoria 3', 300.0, 0);
''')
```

```
## <sqlite3.Cursor object at 0x000001EF5ECA11C0>
```

```
conn.commit()
```

Consultando a tabela de produtos

```
prods = pl.read_database("SELECT * FROM produtos", conn)
print(prods)
```

```
## shape: (5, 5)
##
##   id    nome        categoria     preco   estoque
##   ---   ---         ---           ---     ---
##   i64   str         str           f64     i64
##
##   1     Produto A   Categoria 1   100.0   50
##   2     Produto B   Categoria 2   150.0   30
##   3     Produto C   Categoria 1   200.0   20
##   4     Produto D   Categoria 2   250.0   10
##   5     Produto E   Categoria 3   300.0   0
##
```

JOINs de vendas e produtos

- A coluna **valor** em **vendas** representa o valor de venda do respectivo produto.
- A coluna **preco** em **produtos** representa o valor de compra do respectivo produto.
- Apresente uma tabela com o nome do produto, seu valor de compra e venda, além do lucro no momento da venda.

```
lucros = pl.read_database("""
SELECT produto, valor AS compra, preco AS venda, preco-valor AS lucro
FROM vendas
INNER JOIN produtos ON vendas.produto = produtos.nome
""", conn)
print(lucros)
```

```
## shape: (5, 4)
##
##   produto     compra   venda   lucro
##   ---         ---      ---     ---
##   str         f64      f64     f64
##
##   Produto A   120.5    100.0   -20.5
##   Produto B   200.0    150.0   -50.0
##   Produto C   150.0    200.0   50.0
##   Produto A   300.0    100.0   -200.0
##   Produto C   100.0    200.0   100.0
##
```

Exemplo:

- Qual foi o lucro médio por vendedor?

```
lucro_medio = pl.read_database("""
SELECT vendedor, produto, AVG(preco-valor) AS lucro_medio
FROM vendas
INNER JOIN produtos ON vendas.produto = produtos.nome
GROUP BY vendedor
""", conn)
print(lucro_medio)
```

```
## shape: (3, 3)
##
##   vendedor   produto     lucro_medio
##   ---        ---         ---
##   str        str         f64
##
##   Ana        Produto A   14.75
##   Bruno      Produto A   -200.0
##   Carlos     Produto B   25.0
##
```

```
# FECHAMENTO EXPLÍCITO ANTES DE ENCERRAR O AMBIENTE
cursor.close()
conn.close()
print("Conexão fechada - ambiente pode ser removido com segurança")
```

```
## Conexão fechada - ambiente pode ser removido com segurança
```

```
## Encerra sessão
   virtualenv_remove(venv_python)
```

```
## Virtual environment "temp.venv" removed.
```