Exercício de Banco de Dados

Acessando um Banco de Dados através do Python

>>Em uma **primeira instância**, **criamos** o container do **Docker**, usando o comando:

```
docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=root -e
MYSQL_DATABASE=BD_AVIOES -p 3306:3306 -d mysql:latest
```

>>**Entre** no **Container MySQL** para executar os comandos de criação de tabelas, usando o comando:

```
docker exec -it mysql-container mysql -uroot -proot
```

>>**Crie**, nessa mesma instância, o **banco de dados** e a tabela TB_CLIENTES, usando o comando MySQL:

```
USE BD_AVIOES;

CREATE TABLE TB_CLIENTES (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL
);
```

>>**Criamos** uma **segunda instância** no **PlayWithDocker**, e criamos um **ambiente virtual** do **python**, usando os comandos:

```
python -m venv myenv
```

>>**Entramos** nela, usando:

```
source myenv/bin/activate
```

>>Depois, **instalamos** o **Conector Python**, usando os comandos:

```
pip install mysql-connector-python
```

>>Depois, **criamos** o arquivo **app.py**:

```
vi app.py
```

>>E o **editamos**, colocando esses comandos, lembre-se de **colocar o ip da primeira instância criada na variável host**, veja:

```
import mysql.connector
```

```
from mysql.connector import Error

def create_connection():
    """Cria uma conexão com o banco de dados MySQL."""
    connection = None
    try:
        connection = mysql.connector.connect(
            host='COLOQUE AQUI O IP DA PRIMEIRA INSTÂNCIA CRIADA!!!
(ex.: 192.168.0.12)',
            port='3306',
            user='root',
            password='root',
            database='BD_AVIOES'
        )
        print("Conexão com o MySQL bem-sucedida")
    except Error as e:
        print(f"Erro '{e}' ocorreu")

    return connection

def create_cliente(connection, nome, email):
    """Insere um novo cliente na tabela TB_CLIENTES."""
    cursor = connection.cursor()
    query = "INSERT INTO TB_CLIENTES (nome, email) VALUES (%s, %s)"
    cursor.execute(query, (nome, email))
    connection.commit()
    print("Cliente adicionado com sucesso")

def read_clientes(connection):
    """Lê todos os clientes da tabela TB_CLIENTES."""
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM TB_CLIENTES")
    clientes = cursor.fetchall()
    for cliente in clientes:
        print(cliente)

def update_cliente(connection, cliente_id, nome, email):
    """Atualiza um cliente existente na tabela TB_CLIENTES."""
    cursor = connection.cursor()
    query = "UPDATE TB_CLIENTES SET nome = %s, email = %s WHERE id =
%s"
    cursor.execute(query, (nome, email, cliente_id))
    connection.commit()
```

```
    print("Cliente atualizado com sucesso")

def delete_cliente(connection, cliente_id):
    """Deleta um cliente da tabela TB_CLIENTES."""
    cursor = connection.cursor()
    query = "DELETE FROM TB_CLIENTES WHERE id = %s"
    cursor.execute(query, (cliente_id,))
    connection.commit()
    print("Cliente deletado com sucesso")

def main():
    connection = create_connection()
    if connection is None:
        return

    # Exemplo de uso das funções CRUD
    create_cliente(connection, 'João Silva', 'joao@example.com')
    read_clientes(connection)
    update_cliente(connection, 1, 'João Silva',
'joao.silva@example.com')
    read_clientes(connection)
    delete_cliente(connection, 1)
    read_clientes(connection)

    connection.close()

if __name__ == "__main__":
    main()
```

>>**Execute** o arquivo que você criou:

python app.py

>>É isso! :)