Atividade 08 de BD

Alunos: João Gabriel Tito de Moraes e Maria Clara Ribeiro.

Criando o Banco de Dados e suas Tabelas:

```
##############################################################
#                          WARNING!!!!                       #
# This is a sandbox environment. Using personal credentials  #
# is HIGHLY! discouraged. Any consequences of doing so are   #
# completely the user's responsibilites.                     #
#                                                            #
# The PWD team.                                              #
##############################################################
[node1] (local) root@192.168.0.18 ~
$ docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=BD_AVIOES -p 3306:3306 -d mysql:latest
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
43759093d4f6: Pull complete
d255dceb9ed5: Pull complete
23d22e42ea50: Pull complete
431b106548a3: Pull complete
2be0d473cadf: Pull complete
f56a22f949f9: Pull complete
277ab5f6ddde: Pull complete
df1ba1ac457a: Pull complete
cc9646b08259: Pull complete
893b018337e2: Pull complete
Digest: sha256:146682692a3aa409eae7b7dc6a30f637c6cb49b6ca901c2cd160becc81127d3b
Status: Downloaded newer image for mysql:latest
6e56e48aac780f18b84365fd080987542e1d3e76b9772514244bc6d1aca78d58
[node1] (local) root@192.168.0.18 ~
$ docker exec -it mysql-container mysql -uroot -proot
```

```
mysql> USE BD_AVIOES
Database changed
mysql> CREATE TABLE `TB_CLIENTES` (   `id` varchar(80) NOT NULL PRIMARY KEY,   `customerName` varchar(100) DEFAULT NULL,   `contactLastName` varcha
r(60) DEFAULT NULL,   `contactFirstName` varchar(40) DEFAULT NULL,   `phone` varchar(20) DEFAULT NULL,   `addressLine1` varchar(40) DEFAULT NULL,
 `addressLine2` varchar(40) DEFAULT NULL,   `city` varchar(80) DEFAULT NULL,   `state` varchar(80) DEFAULT NULL,   `postalCode` varchar(60) DEFAULT
 NULL,   `country` varchar(80) DEFAULT NULL,   `salesRepEmployeeNumber` varchar(40) DEFAULT NULL,   `creditLimit` int DEFAULT NULL );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE `TB_DETALHES_PEDIDOS` (
    ->   `id_orderNumber` int NOT NULL,
    ->   `id_productCode` int NOT NULL,
    ->   `quantityOrdered` int DEFAULT NULL,
    ->   `priceEach` int DEFAULT NULL,
    ->   `orderLineNumber` varchar(40) DEFAULT NULL
    -> )
    -> ;
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE `TB_EMPREGADOS` (
    ->   `id` int NOT NULL,
    ->   `lastName` varchar(80) DEFAULT NULL,
    ->   `firstName` varchar(40) DEFAULT NULL,
    ->   `extension` varchar(20) DEFAULT NULL,
    ->   `email` varchar(60) DEFAULT NULL,
    ->   `officeCode` varchar(40) DEFAULT NULL,
    ->   `reportsTo` varchar(40) DEFAULT NULL,
    ->   `jobTittle` varchar(40) DEFAULT NULL
    -> );
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> CREATE TABLE `TB_EMPREGADOS` (
    ->   `id` int NOT NULL,
    ->   `lastName` varchar(80) DEFAULT NULL,
    ->   `firstName` varchar(40) DEFAULT NULL,
    ->   `extension` varchar(20) DEFAULT NULL,
    ->   `email` varchar(60) DEFAULT NULL,
    ->   `officeCode` varchar(40) DEFAULT NULL,
    ->   `reportsTo` varchar(40) DEFAULT NULL,
    ->   `jobTittle` varchar(40) DEFAULT NULL
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE `TB_ESCRITORIOS` (
    ->   `id` int NOT NULL,
    ->   `city` varchar(100) DEFAULT NULL,
    ->   `phone` varchar(40) DEFAULT NULL,
    ->   `addresLine1` varchar(60) DEFAULT NULL,
    ->   `addresLine2` varchar(60) DEFAULT NULL,
    ->   `state` varchar(80) DEFAULT NULL,
    ->   `country` varchar(80) DEFAULT NULL,
    ->   `postalCode` varchar(40) DEFAULT NULL,
    ->   `territory` varchar(40) DEFAULT NULL
    -> ) ;
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE `TB_LINHAS_DE_PRODUTO` (
    ->   `id` int NOT NULL,
    ->   `htmlDescription` text,
    ->   `image` varchar(100) DEFAULT NULL,
```

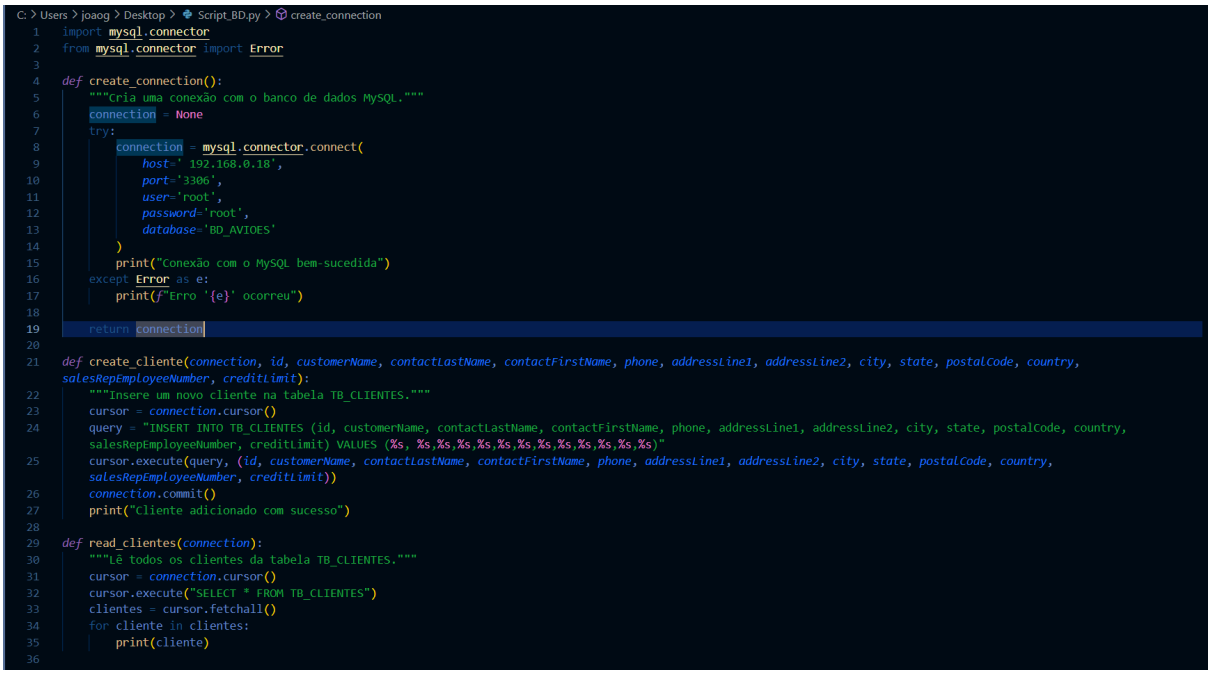
```
    ->   `image` varchar(100) DEFAULT NULL,
    ->   `textDescription` varchar(80) DEFAULT NULL
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE `TB_PAGAMENTOS` (
    ->   `id` int NOT NULL,
    ->   `id_costumerNumber` varchar(80) DEFAULT NULL,
    ->   `paymentDate` varchar(40) DEFAULT NULL,
    ->   `amount` int DEFAULT NULL
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE `TB_PEDIDOS` (
    ->   `id` int NOT NULL,
    ->   `orderDate` varchar(40) DEFAULT NULL,
    ->   `requiredDate` varchar(40) DEFAULT NULL,
    ->   `shippedDate` varchar(40) DEFAULT NULL,
    ->   `status` varchar(40) DEFAULT NULL,
    ->   `comments` text,
    ->   `id_costumerNumber` varchar(80) DEFAULT NULL
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE `TB_PRODUTOS` (
    ->   `id` int NOT NULL,
    ->   `productName` varchar(100) DEFAULT NULL,
    ->   `id_productLine` int DEFAULT NULL,
    ->   `productScale` varchar(100) DEFAULT NULL,
    ->   `productVendor` varchar(100) DEFAULT NULL,
    ->   `productDescription` text,
```

```
mysql> CREATE TABLE `TB_PRODUTOS` (
    ->   `id` int NOT NULL,
    ->   `productName` varchar(100) DEFAULT NULL,
    ->   `id_productLine` int DEFAULT NULL,
    ->   `productScale` varchar(100) DEFAULT NULL,
    ->   `productVendor` varchar(100) DEFAULT NULL,
    ->   `productDescription` text,
    ->   `quantityInStock` int DEFAULT NULL,
    ->   `buyPrice` int DEFAULT NULL,
    ->   `MSRP` int DEFAULT NULL
    -> ) ;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

Criando Ambiente Virtual para Python:

```
################################################################
#                          WARNING!!!!                         #
# This is a sandbox environment. Using personal credentials    #
# is HIGHLY! discouraged. Any consequences of doing so are     #
# completely the user's responsibilites.                       #
#                                                              #
# The PWD team.                                                #
################################################################
[node2] (local) root@192.168.0.17 ~
$ python -m venv myenv
[node2] (local) root@192.168.0.17 ~
$ source myenv/bin/activate
(myenv) [node2] (local) root@192.168.0.17 ~
$ pip install mysql-connector-python
Collecting mysql-connector-python
  Downloading mysql_connector_python-9.2.0-py2.py3-none-any.whl.metadata (6.0 kB)
Downloading mysql_connector_python-9.2.0-py2.py3-none-any.whl (398 kB)
Installing collected packages: mysql-connector-python
Successfully installed mysql-connector-python-9.2.0

[notice] A new release of pip is available: 24.2 -> 25.0.1
[notice] To update, run: pip install --upgrade pip
```

Criando script.py (Prints do VSCode):



```python
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=' 192.168.0.18',
            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, id, customerName, contactLastName, contactFirstName, phone, addressLine1, addressLine2, city, state, postalCode, country, salesRepEmployeeNumber, creditLimit):
    """Insere um novo cliente na tabela TB_CLIENTES."""
    cursor = connection.cursor()
    query = "INSERT INTO TB_CLIENTES (id, customerName, contactLastName, contactFirstName, phone, addressLine1, addressLine2, city, state, postalCode, country, salesRepEmployeeNumber, creditLimit) VALUES (%s, %s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"
    cursor.execute(query, (id, customerName, contactLastName, contactFirstName, phone, addressLine1, addressLine2, city, state, postalCode, country, salesRepEmployeeNumber, creditLimit))
    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)
```

```python
def update_cliente(connection, id, customerName, phone):
    """Atualiza um cliente existente na tabela TB_CLIENTES."""
    cursor = connection.cursor()
    query = "UPDATE TB_CLIENTES SET customerName = %s, phone = %s WHERE id = %s"
    cursor.execute(query, (customerName, phone, id))
    connection.commit()
    print("Cliente atualizado com sucesso")

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

def create_detalhe_pedido(connection, id_orderNumber, id_productCode,quantityOrdered,priceEach,orderLineNumber):
    """Insere um novo detalhe pedido na tabela TB_DETALHES_PEDIDOS."""
    cursor = connection.cursor()
    query = "INSERT INTO TB_DETALHES_PEDIDOS (id_orderNumber, id_productCode, quantityOrdered, priceEach, orderLineNumber) VALUES (%s, %s, %s, %s, %s)"
    cursor.execute(query, (id_orderNumber, id_productCode,quantityOrdered,priceEach,orderLineNumber))
    connection.commit()
    print("Detalhe pedido adicionado com sucesso")

def read_detalhe_pedido(connection):
    """Lê todos os detalhes da tabela TB_DETALHES_PEDIDOS."""
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM TB_DETALHES_PEDIDOS")
    detalhes = cursor.fetchall()
    for detalhe in detalhes:
        print(detalhe)

def update_detalhe_pedido(connection, id_orderNumber, priceEach, quantityOrdered):
    """Atualiza um cliente existente na tabela TB_CLIENTES."""
    cursor = connection.cursor()
    query = "UPDATE TB_DETALHES_PEDIDOS SET priceEach = %s, quantityOrdered = %s WHERE id_orderNumber = %s"
    cursor.execute(query, (priceEach, quantityOrdered, id_orderNumber))
    connection.commit()
    print("Pedido atualizado com sucesso")
```

0 ⚠ 0 Ln 19, Col 22 Spaces: 4 UTF-8 CRLF {} Python 3.12.6 64-bit Go Live

```python
def delete_detalhe_pedido(connection, id_orderNumber):
    """Deleta um cliente da tabela TB_DETALHES_PEDIDOS."""
    cursor = connection.cursor()
    query = "DELETE FROM TB_DETALHES_PEDIDOS WHERE id_orderNumber = %s"
    cursor.execute(query, (id_orderNumber,))
    connection.commit()
    print("Detalhe pedido deletado com sucesso")


def create_empregados(connection, id, lastName,firstName,extension,email, officeCode, reportsTo, jobTittle):
    """insere um novo detalhe pedido na tabela TB_EMPREGADOS."""
    cursor = connection.cursor()
    query = "INSERT INTO TB_EMPREGADOS (id, lastName,firstName,extension,email, officeCode, reportsTo, jobTittle) VALUES (%s, %s, %s, %s,%s,%s, %s, %s)"
    cursor.execute(query, (id, lastName,firstName,extension,email, officeCode, reportsTo, jobTittle))
    connection.commit()
    print("Empregado adicionado com sucesso")

def read_empregados(connection):
    """Lê todos os detalhes da tabela TB_EMPREGADOS."""
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM TB_EMPREGADOS")
    empregados = cursor.fetchall()
    for empregado in empregados:
        print(empregado)

def update_empregado(connection, id, firstName, lastName):
    """Atualiza um cliente existente na tabela TB_EMPREGADOS."""
    cursor = connection.cursor()
    query = "UPDATE TB_EMPREGADOS SET firstName = %s, lastName = %s WHERE id = %s"
    cursor.execute(query, (firstName, lastName, id))
    connection.commit()
    print("Empregado atualizado com sucesso")

def delete_empregado(connection, id):
    """Deleta um cliente da tabela TB_EMPREGADOS."""
    cursor = connection.cursor()
    query = "DELETE FROM TB_EMPREGADOS WHERE id = %s"
    cursor.execute(query, (id,))
    connection.commit()
```



```python
def create_escritorios(connection, id, city,phone,addresline1,addresline2, state, country, postalCode, territory):
    """Insere um novo detalhe pedido na tabela TB_ESCRITORIOS."""
    cursor = connection.cursor()
    query = "INSERT INTO TB_ESCRITORIOS (id, city,phone,addresLine1,addresLine2, state, country, postalCode, territory) VALUES (%s, %s, %s, %s,%s,%s, %s, %s, %s)"
    cursor.execute(query, (id, city,phone,addresline1,addresline2, state, country, postalCode, territory))
    connection.commit()
    print("Escritorio adicionado com sucesso")

def read_escritorios(connection):
    """Lê todos os detalhes da tabela TB_ESCRITORIOS."""
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM TB_ESCRITORIOS")
    escritorios = cursor.fetchall()
    for escritorio in escritorios:
        print(escritorio)

def update_escritorios(connection, id, city, phone):
    """Atualiza um cliente existente na tabela TB_ESCRITORIOS."""
    cursor = connection.cursor()
    query = "UPDATE TB_ESCRITORIOS SET city = %s, phone = %s WHERE id = %s"
    cursor.execute(query, (city, phone, id))
    connection.commit()
    print("Escritorio atualizado com sucesso")

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

def create_linhas_produto(connection, id, htmlDescription,image,textDescription):
    """Insere um novo detalhe pedido na tabela TB_LINHAS_DE_PRODUTO."""
    cursor = connection.cursor()
    query = "INSERT INTO TB_LINHAS_DE_PRODUTO (id, htmlDescription,image,textDescription) VALUES (%s, %s, %s, %s)"
    cursor.execute(query, (id, htmlDescription,image,textDescription))
    connection.commit()
    print("linhas de produto adicionado com sucesso")
```

```python
def read_linhas_produto(connection):
    """Lê todos os detalhes da tabela TB_LINHAS_DE_PRODUTO."""
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM TB_LINHAS_DE_PRODUTO")
    linhas = cursor.fetchall()
    for linha in linhas:
        print(linha)

def update_linhas_produto(connection, id, htmlDescription, image):
    """Atualiza um cliente existente na tabela TB_LINHAS_DE_PRODUTO."""
    cursor = connection.cursor()
    query = "UPDATE TB_LINHAS_DE_PRODUTO SET htmlDescription = %s, image = %s WHERE id = %s"
    cursor.execute(query, (htmlDescription, image, id))
    connection.commit()
    print("Linhas atualizado com sucesso")

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

def create_pagamentos(connection, id, id_costumerNumber,paymentDate,amount):
    """Insere um novo detalhe pedido na tabela TB_PAGAMENTOS."""
    cursor = connection.cursor()
    query = "INSERT INTO TB_PAGAMENTOS (id, id_costumerNumber,paymentDate,amount) VALUES (%s, %s, %s, %s)"
    cursor.execute(query, (id, id_costumerNumber,paymentDate,amount))
    connection.commit()
    print("pagamentos de produto adicionado com sucesso")

def read_pagamentos(connection):
    """Lê todos os detalhes da tabela TB_PAGAMENTOS."""
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM TB_PAGAMENTOS")
    pagamentos = cursor.fetchall()
    for pagamento in pagamentos:
        print(pagamento)
```



```python
def update_pagamentos(connection, id, paymentDate, amount):
    """Atualiza um cliente existente na tabela TB_PAGAMENTOS."""
    cursor = connection.cursor()
    query = "UPDATE TB_PAGAMENTOS SET paymentDate = %s, amount = %s WHERE id = %s"
    cursor.execute(query, (paymentDate, amount, id))
    connection.commit()
    print("pagamento atualizado com sucesso")

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

def create_pedidos(connection, id, orderDate,requiredDate,shippedDate, status, comments, id_costumerNumber):
    """Insere um novo detalhe pedido na tabela TB_PEDIDOS."""
    cursor = connection.cursor()
    query = "INSERT INTO TB_PEDIDOS (id, orderDate,requiredDate,shippedDate, status, comments, id_costumerNumber) VALUES (%s, %s, %s, %s, %s, %s, %s)"
    cursor.execute(query, (id, orderDate,requiredDate,shippedDate, status, comments, id_costumerNumber))
    connection.commit()
    print("pedido de produto adicionado com sucesso")

def read_pedidos(connection):
    """Lê todos os detalhes da tabela TB_PEDIDOS."""
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM TB_PEDIDOS")
    pedidos = cursor.fetchall()
    for pedido in pedidos:
        print(pedido)

def update_pedidos(connection, id, orderDate, status):
    """Atualiza um cliente existente na tabela TB_PEDIDOS."""
    cursor = connection.cursor()
    query = "UPDATE TB_PEDIDOS SET orderDate = %s, status = %s WHERE id = %s"
    cursor.execute(query, (orderDate, status, id))
    connection.commit()
    print("pedido atualizado com sucesso")
```

```python
def delete_pedidos(connection, id):
    """Deleta um cliente da tabela TB_PEDIDOS."""
    cursor = connection.cursor()
    query = "DELETE FROM TB_PEDIDOS WHERE id = %s"
    cursor.execute(query, (id,))
    connection.commit()
    print("pedido deletado com sucesso")

def create_produtos(connection, id, productName,id_productLine,productScale, productVendor, productDescription, quantityInStock, buyPrice, MSRP):
    """Insere um novo detalhe pedido na tabela TB_PRODUTOS."""
    cursor = connection.cursor()
    query = "INSERT INTO TB_PRODUTOS (id, productName,id_productLine,productScale, productVendor, productDescription, quantityInStock, buyPrice, MSRP) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"
    cursor.execute(query, (id, productName,id_productLine,productScale, productVendor, productDescription, quantityInStock, buyPrice, MSRP))
    connection.commit()
    print("produto adicionado com sucesso")

def read_produtos(connection):
    """Lê todos os detalhes da tabela TB_PRODUTOS."""
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM TB_PRODUTOS")
    produtos = cursor.fetchall()
    for produto in produtos:
        print(produto)

def update_produtos(connection, id, productName, buyPrice):
    """Atualiza um cliente existente na tabela TB_PRODUTOS."""
    cursor = connection.cursor()
    query = "UPDATE TB_PRODUTOS SET productName = %s, buyPrice = %s WHERE id = %s"
    cursor.execute(query, (productName, buyPrice, id))
    connection.commit()
    print("produto atualizado com sucesso")

def delete_produtos(connection, id):
    """Deleta um cliente da tabela TB_PRODUTOS."""
    cursor = connection.cursor()
    query = "DELETE FROM TB_PRODUTOS WHERE id = %s"
    cursor.execute(query, (id,))
    connection.commit()
    print("produto deletado com sucesso")
```

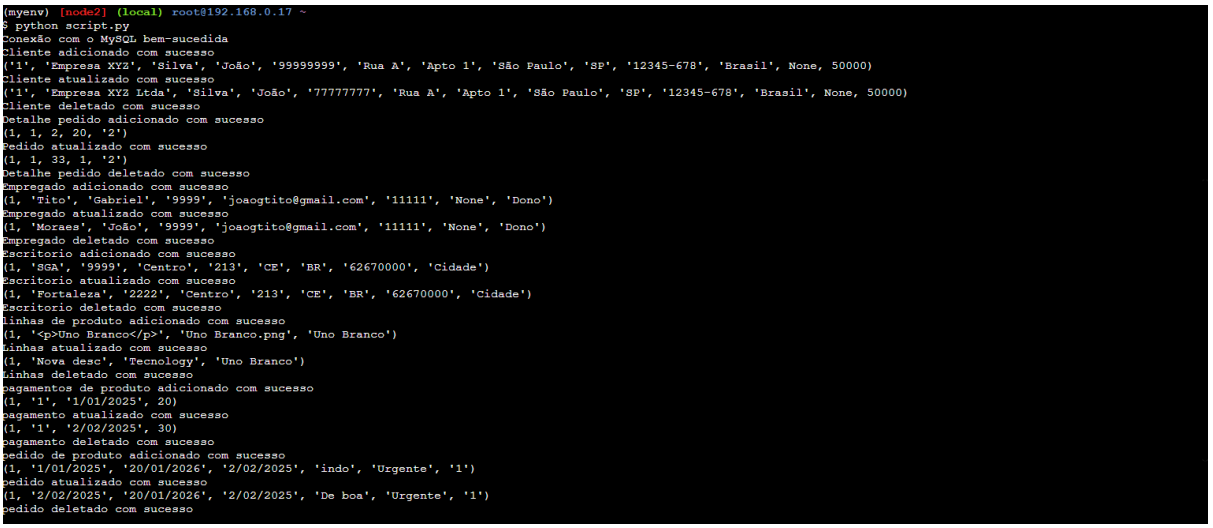
```python
def main():
    connection = create_connection()
    if connection is None:
        return

    # Exemplo de uso das funções CRUD
    create_cliente(connection, 1, 'Empresa XYZ', 'Silva', 'João', '99999999', 'Rua A', 'Apto 1', 'São Paulo', 'SP', '12345-678', 'Brasil', None, 50000.00)
    read_clientes(connection)
    update_cliente(connection, 1, 'Empresa XYZ Ltda', '77777777')
    read_clientes(connection)
    delete_cliente(connection, 1)
    read_clientes(connection)

    #Crud em TB_DETALHES_PEDIDOS
    create_detalhe_pedido(connection, 1, 1, 2, 20, '2')
    read_detalhe_pedido(connection)
    update_detalhe_pedido(connection, 1, 1, 33)
    read_detalhe_pedido(connection)
    delete_detalhe_pedido(connection, 1)
    read_detalhe_pedido(connection)

    #Crud em TB_EMPREGADOS
    create_empregados(connection, 1, 'Tito', 'Gabriel', '9999', 'joaogtito@gmail.com', '11111', 'None', 'Dono')
    read_empregados(connection)
    update_empregado(connection, 1, 'João', 'Moraes')
    read_empregados(connection)
    delete_empregado(connection, 1)
    read_empregados(connection)

    #Crud em TB_ESCRITORIOS
    create_escritorios(connection, 1, 'SGA', '9999', 'Centro', '213', 'CE', 'BR', '62670000', 'Cidade')
    read_escritorios(connection)
    update_escritorios(connection, 1, 'Fortaleza', '2222')
    read_escritorios(connection)
    delete_escritorios(connection, 1)
    read_escritorios(connection)

    #Crud em TB_LINHAS_DE_PRODUTO
    create_linhas_produto(connection, 1, '<p>Uno Branco</p>', 'Uno Branco.png', 'Uno Branco')
```



```python
    #Crud em TB_LINHAS_DE_PRODUTO
    create_linhas_produto(connection, 1, '<p>Uno Branco</p>', 'Uno Branco.png', 'Uno Branco')
    read_linhas_produto(connection)
    update_linhas_produto(connection, 1, 'Nova desc', 'Tecnology')
    read_linhas_produto(connection)
    delete_linhas_produto(connection, 1)
    read_linhas_produto(connection)

    #Crud em TB_PAGAMENTOS
    create_pagamentos(connection, 1, 1, '1/01/2025', 20)
    read_pagamentos(connection)
    update_pagamentos(connection, 1, '2/02/2025', 30)
    read_pagamentos(connection)
    delete_pagamentos(connection, 1)
    read_pagamentos(connection)

    #Crud em TB_PEDIDOS
    create_pedidos(connection, 1, '1/01/2025', '20/01/2026','2/02/2025', 'indo', 'Urgente', 1)
    read_pedidos(connection)
    update_pedidos(connection, 1, '2/02/2025', 'De boa')
    read_pedidos(connection)
    delete_pedidos(connection, 1)
    read_pedidos(connection)

    #Crud em TB_PRODUTOS
    create_produtos(connection, 1, 'Mouse', 1,'1:10', 'Razer', 'Mouse gamer', 1, 20, 150)
    read_produtos(connection)
    update_produtos(connection, 1, 'Musse', 3000)
    read_produtos(connection)
    delete_produtos(connection, 1)
    read_produtos(connection)

    connection.close()

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


(Script no Github)

Saída do Script no Venv:



Arquivo do Banco de Dados usados e Script.py presentes no Github.