

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 responsibilities.  
#  
# The FWD 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  
df1balac457a: Pull complete  
cc9646b08259: Pull complete  
893b018337e2: Pull complete  
Digest: sha256:146682692a3aa409eae7b7dc6a30f637c6cb49b6ca901c2cd160becc81127d3b  
Status: Downloaded newer image for mysql:latest  
6e56e48aac780f18b84365fd080987542e1d3e76b9772514244bc6dlaca78d58  
[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` varchar(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,  
-> `jobTitle` 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,  
-> `jobTitle` 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,  
-> `addressLine1` varchar(60) DEFAULT NULL,  
-> `addressLine2` 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,
-> `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 responsibilities. #
# #
# 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):

```
C:\Users\joaog > Desktop > Script_BD.py > create_connection

1 import mysql.connector
2 from mysql.connector import Error
3
4 def create_connection():
5     """Cria uma conexão com o banco de dados MySQL."""
6     connection = None
7     try:
8         connection = mysql.connector.connect(
9             host='192.168.0.18',
10            port='3306',
11            user='root',
12            password='root',
13            database='BD_AVIOES'
14        )
15        print("Conexão com o MySQL bem-sucedida")
16    except Error as e:
17        print(f"Erro '{e}' ocorreu")
18
19    return connection
20
21 def create_cliente(connection, id, customerName, contactLastName, contactFirstName, phone, addressLine1, addressLine2, city, state, postalCode, country,
22 salesRepEmployeeNumber, creditLimit):
23     """Insere um novo cliente na tabela TB_CLIENTES."""
24     cursor = connection.cursor()
25     query = "INSERT INTO TB_CLIENTES (id, customerName, contactLastName, contactFirstName, phone, addressLine1, addressLine2, city, state, postalCode, country,
26 salesRepEmployeeNumber, creditLimit) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
27     cursor.execute(query, (id, customerName, contactLastName, contactFirstName, phone, addressLine1, addressLine2, city, state, postalCode, country,
28 salesRepEmployeeNumber, creditLimit))
29     connection.commit()
30     print("cliente adicionado com sucesso")
31
32 def read_clientes(connection):
33     """Lê todos os clientes da tabela TB_CLIENTES."""
34     cursor = connection.cursor()
35     cursor.execute("SELECT * FROM TB_CLIENTES")
36     clientes = cursor.fetchall()
37     for cliente in clientes:
38         print(cliente)
39
40 def update_cliente(connection, id, customerName, phone):
41     """Atualiza um cliente existente na tabela TB_CLIENTES."""
42     cursor = connection.cursor()
43     query = "UPDATE TB_CLIENTES SET customerName = %s, phone = %s WHERE id = %s"
44     cursor.execute(query, (customerName, phone, id))
45     connection.commit()
46     print("cliente atualizado com sucesso")
47
48 def delete_cliente(connection, id):
49     """Deleta um cliente da tabela TB_CLIENTES."""
50     cursor = connection.cursor()
51     query = "DELETE FROM TB_CLIENTES WHERE id = %s"
52     cursor.execute(query, (id,))
53     connection.commit()
54     print("cliente deletado com sucesso")
55
56 def create_detalhe_pedido(connection, id_orderNumber, id_productCode, quantityOrdered, priceEach, orderLineNumber):
57     """Insere um novo detalhe pedido na tabela TB_DETALHES_PEDIDOS."""
58     cursor = connection.cursor()
59     query = "INSERT INTO TB_DETALHES_PEDIDOS (id_orderNumber, id_productCode, quantityOrdered, priceEach, orderLineNumber) VALUES (%s, %s, %s, %s, %s)"
60     cursor.execute(query, (id_orderNumber, id_productCode, quantityOrdered, priceEach, orderLineNumber))
61     connection.commit()
62     print("Detalhe pedido adicionado com sucesso")
63
64 def read_detalhe_pedido(connection):
65     """Lê todos os detalhes da tabela TB_DETALHES_PEDIDOS."""
66     cursor = connection.cursor()
67     cursor.execute("SELECT * FROM TB_DETALHES_PEDIDOS")
68     detalhes = cursor.fetchall()
69     for detalhe in detalhes:
70         print(detalhe)
71
72 def update_detalhe_pedido(connection, id_orderNumber, priceEach, quantityOrdered):
73     """Atualiza um cliente existente na tabela TB_CLIENTES."""
74     cursor = connection.cursor()
75     query = "UPDATE TB_DETALHES_PEDIDOS SET priceEach = %s, quantityOrdered = %s WHERE id_orderNumber = %s"
76     cursor.execute(query, (priceEach, quantityOrdered, id_orderNumber))
77     connection.commit()
78     print("Pedido atualizado com sucesso")
```

```

77 def delete_detalhe_pedido(connection, id_orderNumber):
78     """Deleta um cliente da tabela TB_DETALHES_PEDIDOS."""
79     cursor = connection.cursor()
80     query = "DELETE FROM TB_DETALHES_PEDIDOS WHERE id_orderNumber = %s"
81     cursor.execute(query, (id_orderNumber,))
82     connection.commit()
83     print("Detalhe pedido deletado com sucesso")
84
85
86 def create_empregados(connection, id, lastName, firstName, extension, email, officeCode, reportsTo, jobTitle):
87     """Insere um novo detalhe pedido na tabela TB_EMPREGADOS."""
88     cursor = connection.cursor()
89     query = "INSERT INTO TB_EMPREGADOS (id, lastName, firstName, extension, email, officeCode, reportsTo, jobTitle) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
90     cursor.execute(query, (id, lastName, firstName, extension, email, officeCode, reportsTo, jobTitle))
91     connection.commit()
92     print("Empregado adicionado com sucesso")
93
94 def read_empregados(connection):
95     """Le todos os detalhes da tabela TB_EMPREGADOS."""
96     cursor = connection.cursor()
97     cursor.execute("SELECT * FROM TB_EMPREGADOS")
98     empregados = cursor.fetchall()
99     for empregado in empregados:
100         print(empregado)
101
102 def update_empregado(connection, id, firstName, lastName):
103     """Atualiza um cliente existente na tabela TB_EMPREGADOS."""
104     cursor = connection.cursor()
105     query = "UPDATE TB_EMPREGADOS SET firstName = %s, lastName = %s WHERE id = %s"
106     cursor.execute(query, (firstName, lastName, id))
107     connection.commit()
108     print("Empregado atualizado com sucesso")
109
110 def delete_empregado(connection, id):
111     """Deleta um cliente da tabela TB_EMPREGADOS."""
112     cursor = connection.cursor()
113     query = "DELETE FROM TB_EMPREGADOS WHERE id = %s"
114     cursor.execute(query, (id,))
115     connection.commit()

```

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

```

118 def create_escritorios(connection, id, city, phone, addressLine1, addressLine2, state, country, postalCode, territory):
119     """Insere um novo detalhe pedido na tabela TB_ESCRITORIOS."""
120     cursor = connection.cursor()
121     query = "INSERT INTO TB_ESCRITORIOS (id, city, phone, addressLine1, addressLine2, state, country, postalCode, territory) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"
122     cursor.execute(query, (id, city, phone, addressLine1, addressLine2, state, country, postalCode, territory))
123     connection.commit()
124     print("Escritorio adicionado com sucesso")
125
126 def read_escritorios(connection):
127     """Le todos os detalhes da tabela TB_ESCRITORIOS."""
128     cursor = connection.cursor()
129     cursor.execute("SELECT * FROM TB_ESCRITORIOS")
130     escritorios = cursor.fetchall()
131     for escritorio in escritorios:
132         print(escritorio)
133
134 def update_escritorios(connection, id, city, phone):
135     """Atualiza um cliente existente na tabela TB_ESCRITORIOS."""
136     cursor = connection.cursor()
137     query = "UPDATE TB_ESCRITORIOS SET city = %s, phone = %s WHERE id = %s"
138     cursor.execute(query, (city, phone, id))
139     connection.commit()
140     print("Escritorio atualizado com sucesso")
141
142 def delete_escritorios(connection, id):
143     """Deleta um cliente da tabela TB_ESCRITORIOS."""
144     cursor = connection.cursor()
145     query = "DELETE FROM TB_ESCRITORIOS WHERE id = %s"
146     cursor.execute(query, (id,))
147     connection.commit()
148     print("Escritorio deletado com sucesso")
149
150 def create_linhas_produto(connection, id, htmlDescription, image, textDescription):
151     """Insere um novo detalhe pedido na tabela TB_LINHAS_DE_PRODUTO."""
152     cursor = connection.cursor()
153     query = "INSERT INTO TB_LINHAS_DE_PRODUTO (id, htmlDescription, image, textDescription) VALUES (%s, %s, %s, %s)"
154     cursor.execute(query, (id, htmlDescription, image, textDescription))
155     connection.commit()
156     print("Linhas de produto adicionado com sucesso")

```

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

```

158 def read_linhas_produto(connection):
159     """Lê todos os detalhes da tabela TB_LINHAS_DE_PRODUTO."""
160     cursor = connection.cursor()
161     cursor.execute("SELECT * FROM TB_LINHAS_DE_PRODUTO")
162     linhas = cursor.fetchall()
163     for linha in linhas:
164         print(linha)
165
166 def update_linhas_produto(connection, id, htmlDescription, image):
167     """Atualiza um cliente existente na tabela TB_LINHAS_DE_PRODUTO."""
168     cursor = connection.cursor()
169     query = "UPDATE TB_LINHAS_DE_PRODUTO SET htmlDescription = %s, image = %s WHERE id = %s"
170     cursor.execute(query, (htmlDescription, image, id))
171     connection.commit()
172     print("Linhas atualizado com sucesso")
173
174 def delete_linhas_produto(connection, id):
175     """Deleta um cliente da tabela TB_LINHAS_DE_PRODUTO."""
176     cursor = connection.cursor()
177     query = "DELETE FROM TB_LINHAS_DE_PRODUTO WHERE id = %s"
178     cursor.execute(query, (id,))
179     connection.commit()
180     print("Linhas deletado com sucesso")
181
182 def create_pagamentos(connection, id, id_costumerNumber, paymentDate, amount):
183     """Insere um novo detalhe pedido na tabela TB_PAGAMENTOS."""
184     cursor = connection.cursor()
185     query = "INSERT INTO TB_PAGAMENTOS (id, id_costumerNumber, paymentDate, amount) VALUES (%s, %s, %s, %s)"
186     cursor.execute(query, (id, id_costumerNumber, paymentDate, amount))
187     connection.commit()
188     print("pagamentos de produto adicionado com sucesso")
189
190 def read_pagamentos(connection):
191     """Lê todos os detalhes da tabela TB_PAGAMENTOS."""
192     cursor = connection.cursor()
193     cursor.execute("SELECT * FROM TB_PAGAMENTOS")
194     pagamentos = cursor.fetchall()
195     for pagamento in pagamentos:
196         print(pagamento)

```

Ln 19, Col 22

Spaces: 4 UTF-8 CRLF {} Python 3.12.6 64-bit Go Live

```

198 def update_pagamentos(connection, id, paymentDate, amount):
199     """Atualiza um cliente existente na tabela TB_PAGAMENTOS."""
200     cursor = connection.cursor()
201     query = "UPDATE TB_PAGAMENTOS SET paymentDate = %s, amount = %s WHERE id = %s"
202     cursor.execute(query, (paymentDate, amount, id))
203     connection.commit()
204     print("pagamento atualizado com sucesso")
205
206 def delete_pagamentos(connection, id):
207     """Deleta um cliente da tabela TB_PAGAMENTOS."""
208     cursor = connection.cursor()
209     query = "DELETE FROM TB_PAGAMENTOS WHERE id = %s"
210     cursor.execute(query, (id,))
211     connection.commit()
212     print("pagamento deletado com sucesso")
213
214 def create_pedidos(connection, id, orderDate, requiredDate, shippedDate, status, comments, id_costumerNumber):
215     """Insere um novo detalhe pedido na tabela TB_PEDIDOS."""
216     cursor = connection.cursor()
217     query = "INSERT INTO TB_PEDIDOS (id, orderDate, requiredDate, shippedDate, status, comments, id_costumerNumber) VALUES (%s, %s, %s, %s, %s, %s, %s)"
218     cursor.execute(query, (id, orderDate, requiredDate, shippedDate, status, comments, id_costumerNumber))
219     connection.commit()
220     print("pedido de produto adicionado com sucesso")
221
222 def read_pedidos(connection):
223     """Lê todos os detalhes da tabela TB_PEDIDOS."""
224     cursor = connection.cursor()
225     cursor.execute("SELECT * FROM TB_PEDIDOS")
226     pedidos = cursor.fetchall()
227     for pedido in pedidos:
228         print(pedido)
229
230 def update_pedidos(connection, id, orderDate, status):
231     """Atualiza um cliente existente na tabela TB_PEDIDOS."""
232     cursor = connection.cursor()
233     query = "UPDATE TB_PEDIDOS SET orderDate = %s, status = %s WHERE id = %s"
234     cursor.execute(query, (orderDate, status, id))
235     connection.commit()
236     print("pedido atualizado com sucesso")

```

Ln 19, Col 22

Spaces: 4 UTF-8 CRLF {} Python 3.12.6 64-bit Go Live

```

238 def delete_pedidos(connection, id):
239     """Deleta um cliente da tabela TB_PEDIDOS."""
240     cursor = connection.cursor()
241     query = "DELETE FROM TB_PEDIDOS WHERE id = %s"
242     cursor.execute(query, (id,))
243     connection.commit()
244     print("pedido deletado com sucesso")
245
246 def create_produtos(connection, id, productName, id_productLine, productScale, productVendor, productDescription, quantityInStock, buyPrice, MSRP):
247     """Insere um novo detalhe pedido na tabela TB_PRODUTOS."""
248     cursor = connection.cursor()
249     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)"
250     cursor.execute(query, (id, productName, id_productLine, productScale, productVendor, productDescription, quantityInStock, buyPrice, MSRP))
251     connection.commit()
252     print("produto adicionado com sucesso")
253
254 def read_produtos(connection):
255     """Le todos os detalhes da tabela TB_PRODUTOS."""
256     cursor = connection.cursor()
257     cursor.execute("SELECT * FROM TB_PRODUTOS")
258     produtos = cursor.fetchall()
259     for produto in produtos:
260         print(produto)
261
262 def update_produtos(connection, id, productName, buyPrice):
263     """Atualiza um cliente existente na tabela TB_PRODUTOS."""
264     cursor = connection.cursor()
265     query = "UPDATE TB_PRODUTOS SET productName = %s, buyPrice = %s WHERE id = %s"
266     cursor.execute(query, (productName, buyPrice, id))
267     connection.commit()
268     print("produto atualizado com sucesso")
269
270 def delete_produtos(connection, id):
271     """Deleta um cliente da tabela TB_PRODUTOS."""
272     cursor = connection.cursor()
273     query = "DELETE FROM TB_PRODUTOS WHERE id = %s"
274     cursor.execute(query, (id,))
275     connection.commit()

```

0 0 0

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

```

279 def main():
280     connection = create_connection()
281     if connection is None:
282         return
283
284     # Exemplo de uso das funções CRUD
285     create_cliente(connection, 1, 'Empresa XYZ', 'Silva', 'João', '99999999', 'Rua A', 'Apto 1', 'São Paulo', 'SP', '12345-678', 'Brasil', None, 50000.00)
286     read_clientes(connection)
287     update_cliente(connection, 1, 'Empresa XYZ Ltda', '77777777')
288     read_clientes(connection)
289     delete_cliente(connection, 1)
290     read_clientes(connection)
291
292     #Crud em TB_DETALHES_PEDIDOS
293     create_detalhe_pedido(connection, 1, 1, 2, 20, '2')
294     read_detalhe_pedido(connection)
295     update_detalhe_pedido(connection, 1, 1, 33)
296     read_detalhe_pedido(connection)
297     delete_detalhe_pedido(connection, 1)
298     read_detalhe_pedido(connection)
299
300     #Crud em TB_EMPREGADOS
301     create_empregados(connection, 1, 'Tito', 'Gabriel', '9999', 'joaogtito@gmail.com', '11111', 'None', 'Dono')
302     read_empregados(connection)
303     update_empregado(connection, 1, 'João', 'Moraes')
304     read_empregados(connection)
305     delete_empregado(connection, 1)
306     read_empregados(connection)
307
308     #Crud em TB_ESCRITORIOS
309     create_escritorios(connection, 1, 'SGA', '9999', 'Centro', '213', 'CE', 'BR', '62670000', 'Cidade')
310     read_escritorios(connection)
311     update_escritorios(connection, 1, 'Fortaleza', '2222')
312     read_escritorios(connection)
313     delete_escritorios(connection, 1)
314     read_escritorios(connection)
315
316     #Crud em TB LINHAS DE PRODUTO
317     create_linhas_produto(connection, 1, '<p>Uno Branco</p>', 'Uno Branco.png', 'Uno Branco')

```

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

```

315
316     #Crud em TB LINHAS DE PRODUTO
317     create_linhas_produto(connection, 1, '<p>Uno Branco</p>', 'Uno Branco.png', 'Uno Branco')
318     read_linhas_produto(connection)
319     update_linhas_produto(connection, 1, 'Nova desc', 'Technology')
320     read_linhas_produto(connection)
321     delete_linhas_produto(connection, 1)
322     read_linhas_produto(connection)
323
324     #Crud em TB PAGAMENTOS
325     create_pagamentos(connection, 1, 1, '1/01/2025', 20)
326     read_pagamentos(connection)
327     update_pagamentos(connection, 1, '2/02/2025', 30)
328     read_pagamentos(connection)
329     delete_pagamentos(connection, 1)
330     read_pagamentos(connection)
331
332     #Crud em TB_PEDIDOS
333     create_pedidos(connection, 1, '1/01/2025', '20/01/2026', '2/02/2025', 'Indo', 'Urgente', 1)
334     read_pedidos(connection)
335     update_pedidos(connection, 1, '2/02/2025', 'De boa')
336     read_pedidos(connection)
337     delete_pedidos(connection, 1)
338     read_pedidos(connection)
339
340     #Crud em TB_PRODUTOS
341     create_produtos(connection, 1, 'Mouse', 1, '1:10', 'Razer', 'Mouse gamer', 1, 20, 150)
342     read_produtos(connection)
343     update_produtos(connection, 1, 'Musse', 3000)
344     read_produtos(connection)
345     delete_produtos(connection, 1)
346     read_produtos(connection)
347
348     connection.close()
349
350 if __name__ == "__main__":
351     main()

```

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

(Script no Github)

Saída do Script no Venv:

```
(myenv) [node2] (local) root@192.168.0.17 ~
$ python script.py
Conexão com o MySQL bem-sucedida
Cliente adicionado com sucesso
('1', 'Empresa XYZ', 'Silva', 'João', '99999999', 'Rua A', 'Apto 1', 'São Paulo', 'SP', '12345-678', 'Brasil', None, 50000)
Cliente atualizado com sucesso
('1', 'Empresa XYZ Ltda', 'Silva', 'João', '77777777', 'Rua A', 'Apto 1', 'São Paulo', 'SP', '12345-678', 'Brasil', None, 50000)
Cliente deletado com sucesso
Detalhe pedido adicionado com sucesso
(1, 1, 2, 20, '2')
Pedido atualizado com sucesso
(1, 1, 33, 1, '2')
Detalhe pedido deletado com sucesso
Empregado adicionado com sucesso
(1, 'Rito', 'Gabriel', '9999', 'joaogtito@gmail.com', '11111', 'None', 'Dono')
Empregado atualizado com sucesso
(1, 'Moraes', 'João', '9999', 'joaogtito@gmail.com', '11111', 'None', 'Dono')
Empregado deletado com sucesso
Escritorio adicionado com sucesso
(1, 'SGA', '9999', 'Centro', '213', 'CE', 'BR', '62670000', 'Cidade')
Escritorio atualizado com sucesso
(1, 'Fortaleza', '2222', 'Centro', '213', 'CE', 'BR', '62670000', 'Cidade')
Escritorio deletado com sucesso
Linhas de produto adicionado com sucesso
(1, '<p>Uno Branco</p>', 'Uno Branco.png', 'Uno Branco')
Linhas atualizado com sucesso
(1, 'Nova desc', 'Technology', 'Uno Branco')
Linhas deletado com sucesso
Pagamentos de produto adicionado com sucesso
(1, '1', '1/01/2025', 20)
Pagamento atualizado com sucesso
(1, '1', '2/02/2025', 30)
Pagamento deletado com sucesso
Pedido de produto adicionado com sucesso
(1, '1/01/2025', '20/01/2026', '2/02/2025', 'Indo', 'Urgente', '1')
Pedido atualizado com sucesso
(1, '2/02/2025', '20/01/2026', '2/02/2025', 'De boa', 'Urgente', '1')
Pedido deletado com sucesso
```

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