

```
#####
#                                     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.
#####
[node1] (local) root@192.168.0.13 ~
$ ls
[node1] (local) root@192.168.0.13 ~
$ mkdir teste
[node1] (local) root@192.168.0.13 ~
$ ls
teste
[node1] (local) root@192.168.0.13 ~
$ cd teste
[node1] (local) root@192.168.0.13 ~/teste
$ vi docker-compose.yml
[node1] (local) root@192.168.0.13 ~/teste
$ vi python-sql.py
[node1] (local) root@192.168.0.13 ~/teste
$ ls
docker-compose.yml  python-sql.py
[node1] (local) root@192.168.0.13 ~/teste
$ █
```

vi é um editor de texto que te permite editar texto e criar arquivos.
ls é um comando usado pra listar os arquivos em um diretório
cd é um comando usado pra navegar à um diretório específico

Vi

Para poder escrever no vi, digite a tecla “a”, tem outras teclas, mas essa tbm funciona.
Quando você fizer isso, vai entrar no modo de insert, para colar textos que você copiou,
aperte as teclas “control+v” ou “control+shift+v” .

Para salvar e sair do arquivo, você precisa apertar a tecla “esq” para sair do modo de
INSERT e apertar a tecla “.” (dois pontos), após isso digitar “wq” e apertar enter.

Assim você vai ter salvo o conteudo dos seus arquivos.

```
1 services:
2   db:
3     image: mysql:latest
4     restart: always
5     ports:
6       - '3306:3306'
7     environment:
8       MYSQL_ROOT_PASSWORD: 'cayque123'
9       MYSQL_USER: 'cayque'
10      MYSQL_PASSWORD: 'cayque321'
11      MYSQL_DATABASE: 'cayqueBD'
12
13
14   phpmyadmin:
15     image: phpmyadmin:latest
16     ports:
17       - '8080:80'
18     environment:
19       PMA_HOST: db
20       MYSQL_ROOT_PASSWORD: 'cayque123'
21     depends_on:
22       - db
23
```

Serviços representam os contêineres que vão ser executados. Atribua um valor a cada um desses atributos de acordo com o que se deseja.

text

```
version: '3.8'
services:
  db:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: senha_segura
    volumes:
      - mysql_data:/var/lib/mysql

  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    environment:
      PMA_HOST: db
      PMA_PORT: 3306
    ports:
      - 8080:80
    depends_on:
      - db

volumes:
  mysql_data:
```

Configura o arquivo .yaml do seu jeito ou copiando e colando o resultado do prompt da IA. O objetivo de um arquivo .yaml é executar com mais facilidades múltiplos contêineres que dependem de outros contêineres.

```
[node1] (local) root@192.168.0.68 ~/teste
$ vi docker-compose.yml
[node1] (local) root@192.168.0.68 ~/teste
$ docker-compose up -d
```

Executa o comando docker-compose up -d pra inicializar todos os containers descritos no arquivo .yaml

```
[node1] (local) root@192.168.0.68 ~/teste
$ docker-compose up -d
[+] Running 18/2
  ⋮ db [ ] Pulling
  ⋮ phpmyadmin [ ] Pulling
```

O comando 'docker ps' mostra quais contêineres estão rodando atualmente:

```
bash: 0c2RR0: command not found
bash: 276: command not found
bash: 0casdadssa[: command not found
[node1] (local) root@192.168.0.68 ~/teste
$ vi docker-compose.yml
[node1] (local) root@192.168.0.68 ~/teste
$ docker-compose up -d
[+] Running 33/2
  ✓ db Pulled 36.2s
  ✓ phpmyadmin Pulled 21.0s
[+] Running 3/3
  ✓ Network teste_default Created 0.1s
  ✓ Container teste-db-1 Started 0.5s
  ✓ Container teste-phpmyadmin-1 Started 0.8s
[node1] (local) root@192.168.0.68 ~/teste
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORT
f5821bf419ef	phpmyadmin:latest	"/docker-entrypoint.s..."	17 seconds ago	Up 16 seconds	0.0.
0.0:8080->80/tcp		teste-phpmyadmin-1			
fd109a9df751	mysql:latest	"docker-entrypoint.s..."	17 seconds ago	Up 16 seconds	0.0.
0.0:3306->3306/tcp, 33060/tcp		teste-db-1			

```
[node1] (local) root@192.168.0.68 ~/teste
$
```

Agora vamos configurar o arquivo '[python-sql.py](#)'

Primeiramente, a gente precisa usar a biblioteca mysql-connector-python, que precisa ser baixado pelo pip install, entretanto o pip não é normalmente utilizável por meios normais, logo a gente vai seguir essas etapas aqui:

- voltar pro diretório anterior -> 'cd ..'
- transformar o diretório que contém o arquivo .yml e o arquivo .py em um ambiente virtual -> 'python -m venv teste'
- Ativar o ambiente virtual -> 'source bin/activate' se vc estiver dentro do diretório transformado ou 'source pastaQualquer/bin/activate' se vc estiver anterior ao diretório transformado.
- Agora é só rodar o comando -> 'pip install mysql-connector-python' e ele será baixado.

Para usar a biblioteca do mysql-connector-python, você não pode sair do ambiente virtual.

```

[node1] (local) root@192.168.0.68 ~/teste
$ cd ..
[node1] (local) root@192.168.0.68 ~
$ python -m venv teste
[node1] (local) root@192.168.0.68 ~
$ ls
teste
[node1] (local) root@192.168.0.68 ~
$ cd teste
[node1] (local) root@192.168.0.68 ~/teste
$ ls
bin          lib          pyvenv.cfg
docker-compose.yml  lib64
include       python-sql.py
[node1] (local) root@192.168.0.68 ~/teste
$ cd ..
[node1] (local) root@192.168.0.68 ~
$ source teste/bin/activate
(teste) [node1] (local) root@192.168.0.68 ~
$ pip install mysql-connector-python
Collecting mysql-connector-python
  Downloading mysql_connector_python-9.3.0-py2.py3-none-any.whl.metadata (7.3 kB)
  Downloading mysql_connector_python-9.3.0-py2.py3-none-any.whl (399 kB)

```

Agora vamos ver o script de python que o prof gerou com IA. Eu testei algumas coisas nele e tava dando erro de operação de multiplicação, depois de transformar tudo em float e fazer com que o cursor retornasse dicionários ao invés de tuplas, começou a funcionar (solução do ChatGPT, eu não tava nem um pouco a fim de debugar esse código de +200 linhas)

Pra abrir o arquivo python, é só seguir essa sintaxe: vi [arquivo.py](#)

```

1 import mysql.connector
2 from mysql.connector import Error
3
4 # Função para criar conexão com o banco BD_VENDAS
5 def criar_conexao():
6     try:
7         conexao = mysql.connector.connect(
8             host='192.168.0.13',          # ajuste conforme seu ambiente
9             user='root',                  # seu usuário MySQL
10            password='cayque123',         # sua senha MySQL
11            database='cayqueBD',
12            port='3306'
13        )
14        if conexao.is_connected():
15            print("Conexão ao cayqueBD realizada com sucesso!")
16            return conexao
17        except Error as e:
18            print(f"Erro ao conectar ao MySQL: {e}")
19            return None
20
21 # Função para criar as tabelas TB_Produtos e TB_FORNECEDOR
22 def criar_tabelas(conexao):
23     cursor = conexao.cursor()
24
25     # Criar tabela TB_Produtos
26     # valor_estoque será calculado na aplicação, pois MySQL não permite coluna calculada diretamente no decimal (pode usar generated column em versões recentes)
27     sql_produtos = """
28     CREATE TABLE IF NOT EXISTS TB_Produtos (
29         id INT AUTO_INCREMENT PRIMARY KEY,
30         descricao VARCHAR(50) NOT NULL,
31         valor_unitario DECIMAL(10,2) NOT NULL,
32         qtd_estoque DECIMAL(10,2) NOT NULL,

```

Lembre de na função “criar_conexao” colocar o host, user, password, database e port da maneira como você declarou no arquivo docker-compose.yml

```
33         valor_estoque DECIMAL(12,2) NOT NULL
34     )
35     """
36
37     # Criar tabela TB_FORNECEDOR
38     sql_fornecedor = """
39     CREATE TABLE IF NOT EXISTS TB_FORNECEDOR (
40         id INT AUTO_INCREMENT PRIMARY KEY,
41         nome VARCHAR(50) NOT NULL
42     )
43     """
44
45     try:
46         cursor.execute(sql_produtos)
47         cursor.execute(sql_fornecedor)
48         conexao.commit()
49         print("Tabelas criadas com sucesso!")
50     except Error as e:
51         print(f"Erro ao criar tabelas: {e}")
52     finally:
53         cursor.close()
54
55 # Funções CRUD para TB_Produtos
56
57 def inserir_produto(conexao, descricao, valor_unitario, qtd_estoque):
58     cursor = conexao.cursor()
59     valor_estoque = valor_unitario * qtd_estoque
60     sql = """
61     INSERT INTO TB_Produtos (descricao, valor_unitario, qtd_estoque, valor_estoque)
62     VALUES (%s, %s, %s, %s)
63     """
64     try:
65         cursor.execute(sql, (descricao, valor_unitario, qtd_estoque, valor_estoque))
66         conexao.commit()
67         print(f"Produto '{descricao}' inserido com sucesso.")
68     except Error as e:
69         print(f"Erro ao inserir produto: {e}")
70     finally:
71         cursor.close()
72
73 def selecionar_produtos(conexao):
74     cursor = conexao.cursor(dictionary=True)
75     sql = "SELECT * FROM TB_Produtos"
76     try:
77         cursor.execute(sql)
78         resultados = cursor.fetchall()
79         return resultados
80     except Error as e:
81         print(f"Erro ao selecionar produtos: {e}")
82         return []
83     finally:
84         cursor.close()
85
86 def atualizar_produto(conexao, id_produto, descricao=None, valor_unitario=None, qtd_estoque=None):
87     cursor = conexao.cursor(dictionary=True)
88     # Buscar dados atuais para recalcular valor_estoque
89     sql_select = "SELECT valor_unitario, qtd_estoque FROM TB_Produtos WHERE id = %s"
90     try:
91         cursor.execute(sql_select, (id_produto,))
92         produto = cursor.fetchone()
93         if not produto:
```

65,1 15%

```
66         conexao.commit()
67         print(f"Produto '{descricao}' inserido com sucesso.")
68     except Error as e:
69         print(f"Erro ao inserir produto: {e}")
70     finally:
71         cursor.close()
72
73 def selecionar_produtos(conexao):
74     cursor = conexao.cursor(dictionary=True)
75     sql = "SELECT * FROM TB_Produtos"
76     try:
77         cursor.execute(sql)
78         resultados = cursor.fetchall()
79         return resultados
80     except Error as e:
81         print(f"Erro ao selecionar produtos: {e}")
82         return []
83     finally:
84         cursor.close()
85
86 def atualizar_produto(conexao, id_produto, descricao=None, valor_unitario=None, qtd_estoque=None):
87     cursor = conexao.cursor(dictionary=True)
88     # Buscar dados atuais para recalcular valor_estoque
89     sql_select = "SELECT valor_unitario, qtd_estoque FROM TB_Produtos WHERE id = %s"
90     try:
91         cursor.execute(sql_select, (id_produto,))
92         produto = cursor.fetchone()
93         if not produto:
```

62,1 29%

O erro está nessa parte aqui da função “atualizar_produto”, as diferenças do código da IA para esse é:

- Duas novas variáveis foram criadas: “atual_valor_unitario” e “atual_qtd_estoque”
- O tipo das variáveis “atual_valor_unitario”, “atual_qtd_estoque”; “novo_valor_unitario”, “nova_qtd_estoque”, “novo_valor_estoque” é float;
- Na atribuição do “novo_valor_unitario” após a palavra chave “else”, ao invés de produto[0] agora é atual_valor_unitario;
- Na atribuição da “nova_qtd_estoque” após a palavra chave “else”, ao invés de produto[1] agora é “atual_qtd_estoque”;

```
93 |         if not produto:
94 |             print("Produto não encontrado.")
95 |             return
96 |
97 |         atual_valor_unitario = float(produto['valor_unitario'])
98 |         atual_qtd_estoque = float(produto['qtd_estoque'])
99 |
100 |         novo_valor_unitario = float(valor_unitario) if valor_unitario is not None else atual_
valor_unitario
101 |         nova_qtd_estoque = float(qtd_estoque) if qtd_estoque is not None else atual_qtd_esto
que
102 |         novo_valor_estoque = novo_valor_unitario * nova_qtd_estoque
103 |
104 |         # Montar query dinâmica para atualizar somente campos fornecidos
105 |         campos = []
106 |         valores = []
107 |         if descricao is not None:
108 |             campos.append("descricao = %s")
109 |             valores.append(descricao)
110 |         if valor_unitario is not None:
111 |             campos.append("valor_unitario = %s")
112 |             valores.append(valor_unitario)
113 |         if qtd_estoque is not None:
114 |             campos.append("qtd_estoque = %s")
115 |             valores.append(qtd_estoque)
116 |
117 |         # Sempre atualizar valor_estoque
118 |         campos.append("valor_estoque = %s")
119 |         valores.append(novo_valor_estoque)
120 |
121 |         valores.append(id_produto)
122 |
123 |         sql_update = f"UPDATE TB_Produtos SET {' '.join(campos)} WHERE id = %s"
```

93,1

44%

```

124         cursor.execute(sql_update, tuple(valores))
125         conexao.commit()
126         print(f"Produto ID {id_produto} atualizado com sucesso.")
127     except Error as e:
128         print(f"Erro ao atualizar produto: {e}")
129     finally:
130         cursor.close()
131
132 def deletar_produto(conexao, id_produto):
133     cursor = conexao.cursor()
134     sql = "DELETE FROM TB_Produtos WHERE id = %s"
135     try:
136         cursor.execute(sql, (id_produto,))
137         conexao.commit()
138         print(f"Produto ID {id_produto} deletado com sucesso.")
139     except Error as e:
140         print(f"Erro ao deletar produto: {e}")
141     finally:
142         cursor.close()
143
144 # Funções CRUD para TB_FORNECEDOR
145
146 def inserir_fornecedor(conexao, nome):
147     cursor = conexao.cursor()
148     sql = "INSERT INTO TB_FORNECEDOR (nome) VALUES (%s)"
149     try:
150         cursor.execute(sql, (nome,))
151         conexao.commit()
152         print(f"Fornecedor '{nome}' inserido com sucesso.")
153     except Error as e:
154         print(f"Erro ao inserir fornecedor: {e}")
155     finally:
156         cursor.close()

```

156,1

60%

```

157
158 def selecionar_fornecedores(conexao):
159     cursor = conexao.cursor(dictionary=True)
160     sql = "SELECT * FROM TB_FORNECEDOR"
161     try:
162         cursor.execute(sql)
163         resultados = cursor.fetchall()
164         return resultados
165     except Error as e:
166         print(f"Erro ao selecionar fornecedores: {e}")
167     return []
168     finally:
169         cursor.close()
170
171 def atualizar_fornecedor(conexao, id_fornecedor, nome):
172     cursor = conexao.cursor()
173     sql = "UPDATE TB_FORNECEDOR SET nome = %s WHERE id = %s"
174     try:
175         cursor.execute(sql, (nome, id_fornecedor))
176         conexao.commit()
177         print(f"Fornecedor ID {id_fornecedor} atualizado com sucesso.")
178     except Error as e:
179         print(f"Erro ao atualizar fornecedor: {e}")
180     finally:
181         cursor.close()
182
183 def deletar_fornecedor(conexao, id_fornecedor):
184     cursor = conexao.cursor()
185     sql = "DELETE FROM TB_FORNECEDOR WHERE id = %s"
186     try:
187         cursor.execute(sql, (id_fornecedor,))
188         conexao.commit()
189         print(f"Fornecedor ID {id_fornecedor} deletado com sucesso.")

```

189,1

76%


```

190     except Error as e:
191         print(f"Erro ao deletar fornecedor: {e}")
192     finally:
193         cursor.close()
194
195 # Função para calcular o valor total do estoque (soma de valor_estoque de todos os produtos)
196 def calcular_valor_total_estoque(conexao):
197     cursor = conexao.cursor()
198     sql = "SELECT SUM(valor_estoque) FROM TB_Produtos"
199     try:
200         cursor.execute(sql)
201         resultado = cursor.fetchone()
202         valor_total = resultado[0] if resultado[0] is not None else 0
203         print(f"Valor total do estoque: R$ {valor_total:.2f}")
204         return valor_total
205     except Error as e:
206         print(f"Erro ao calcular valor total do estoque: {e}")
207         return 0
208     finally:
209         cursor.close()
210
211 # Exemplo de uso
212 if __name__ == "__main__":
213     conexao = criar_conexao()
214     if conexao:
215         criar_tabelas(conexao)
216
217         # Inserir exemplo
218         inserir_produto(conexao, "Teclado Mecânico", 250.00, 10)
219         inserir_produto(conexao, "Mouse Gamer", 150.00, 20)
220
221         inserir_fornecedor(conexao, "Fornecedor A")
222         inserir_fornecedor(conexao, "Fornecedor B")

```

222 1

924

```

206     print(f"Erro ao calcular valor total do estoque: {e}")
207     return 0
208 finally:
209     cursor.close()
210
211 # Exemplo de uso
212 if __name__ == "__main__":
213     conexao = criar_conexao()
214     if conexao:
215         criar_tabelas(conexao)
216
217         # Inserir exemplo
218         inserir_produto(conexao, "Teclado Mecânico", 250.00, 10)
219         inserir_produto(conexao, "Mouse Gamer", 150.00, 20)
220
221         inserir_fornecedor(conexao, "Fornecedor A")
222         inserir_fornecedor(conexao, "Fornecedor B")
223
224         # Listar produtos
225         produtos = selecionar_produtos(conexao)
226         print("Produtos cadastrados:")
227         for p in produtos:
228             print(p)
229
230         # Atualizar produto
231         atualizar_produto(conexao, 1, valor_unitario=260.00)
232
233         # Calcular valor total do estoque
234         calcular_valor_total_estoque(conexao)
235
236         # Fechar conexão
237         conexao.close()
238

```

238,0-1

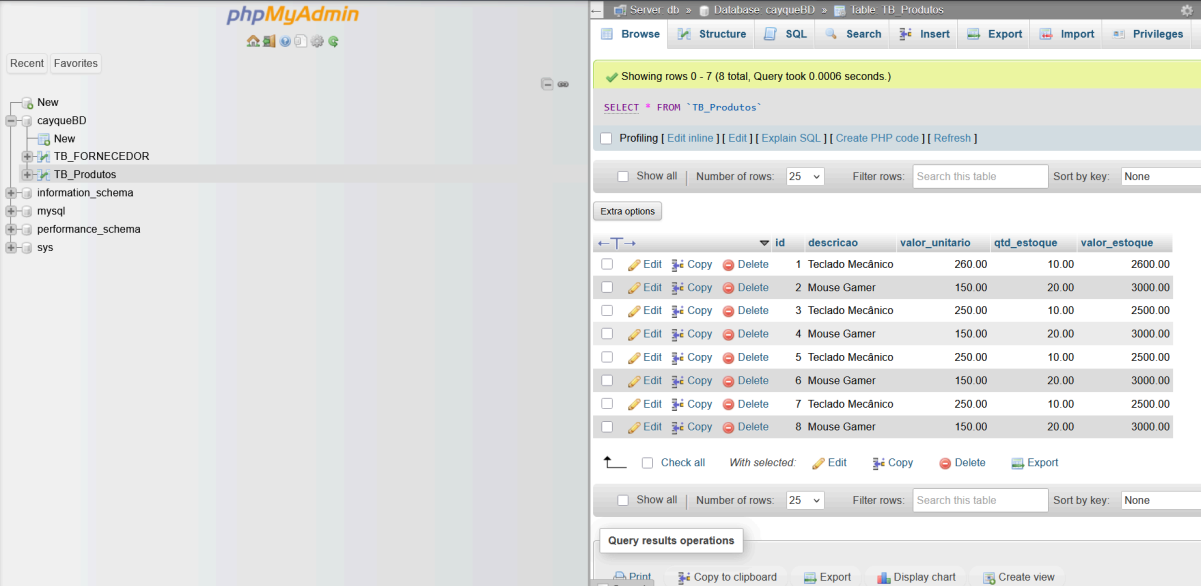
Bot

```
(teste) [node1] (local) root@192.168.0.13 ~/teste
$ python python-sql.py
Conexão ao cayqueBD realizada com sucesso!
Tabelas criadas com sucesso!
Produto 'Teclado Mecânico' inserido com sucesso.
Produto 'Mouse Gamer' inserido com sucesso.
Fornecedor 'Fornecedor A' inserido com sucesso.
Fornecedor 'Fornecedor B' inserido com sucesso.
Produtos cadastrados:
{'id': 1, 'descricao': 'Teclado Mecânico', 'valor_unitario': Decimal('260.00'), 'qtd_estoque': De
cimal('10.00'), 'valor_estoque': Decimal('2600.00')}
{'id': 2, 'descricao': 'Mouse Gamer', 'valor_unitario': Decimal('150.00'), 'qtd_estoque': Decimal
('20.00'), 'valor_estoque': Decimal('3000.00')}
{'id': 3, 'descricao': 'Teclado Mecânico', 'valor_unitario': Decimal('250.00'), 'qtd_estoque': De
cimal('10.00'), 'valor_estoque': Decimal('2500.00')}
{'id': 4, 'descricao': 'Mouse Gamer', 'valor_unitario': Decimal('150.00'), 'qtd_estoque': Decimal
('20.00'), 'valor_estoque': Decimal('3000.00')}
{'id': 5, 'descricao': 'Teclado Mecânico', 'valor_unitario': Decimal('250.00'), 'qtd_estoque': De
cimal('10.00'), 'valor_estoque': Decimal('2500.00')}
{'id': 6, 'descricao': 'Mouse Gamer', 'valor_unitario': Decimal('150.00'), 'qtd_estoque': Decimal
('20.00'), 'valor_estoque': Decimal('3000.00')}
{'id': 7, 'descricao': 'Teclado Mecânico', 'valor_unitario': Decimal('250.00'), 'qtd_estoque': De
cimal('10.00'), 'valor_estoque': Decimal('2500.00')}
{'id': 8, 'descricao': 'Mouse Gamer', 'valor_unitario': Decimal('150.00'), 'qtd_estoque': Decimal
('20.00'), 'valor_estoque': Decimal('3000.00')}
Produto ID 1 atualizado com sucesso.
Valor total do estoque: R$ 22100.00
```

O valor está 22100.00 porque eu executei o arquivo várias vezes.

Para executar um arquivo python siga a sintaxe: python [arquivo.py](#)

Lembrando que como o container do phpmyadmin também está rodando, se você abrir o local host na porta 8080, você vai poder ver mais graficamente o seu banco de dados.



The screenshot shows the phpMyAdmin web interface. On the left is a sidebar with a tree view of the database structure, including 'cayqueBD', 'TB_FORNECEDOR', and 'TB_Produtos'. The main panel displays the 'TB_Produtos' table with a table structure overview and a data grid. The data grid shows 8 rows of product information. Below the table are options for query results operations like 'Print', 'Copy to clipboard', 'Export', 'Display chart', and 'Create view'.

	id	descricao	valor_unitario	qtd_estoque	valor_estoque
<input type="checkbox"/>	1	Teclado Mecânico	260.00	10.00	2600.00
<input type="checkbox"/>	2	Mouse Gamer	150.00	20.00	3000.00
<input type="checkbox"/>	3	Teclado Mecânico	250.00	10.00	2500.00
<input type="checkbox"/>	4	Mouse Gamer	150.00	20.00	3000.00
<input type="checkbox"/>	5	Teclado Mecânico	250.00	10.00	2500.00
<input type="checkbox"/>	6	Mouse Gamer	150.00	20.00	3000.00
<input type="checkbox"/>	7	Teclado Mecânico	250.00	10.00	2500.00
<input type="checkbox"/>	8	Mouse Gamer	150.00	20.00	3000.00