
# Conexão com BD

Durante meu aprendizado de conectar Python com MySQL Workbench, utilizei as seguintes tecnologias:

- Docker compose
- Python
- MySQL Workbench

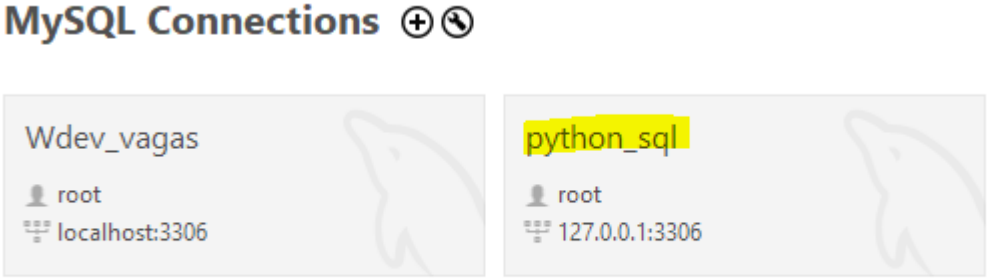
## Docker-Compose

meu código do docker-compose.yml

```
version: "3"
services:
  db:
    image: mariadb
    container_name: python_sql
    restart: always
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: example
```

# SQL

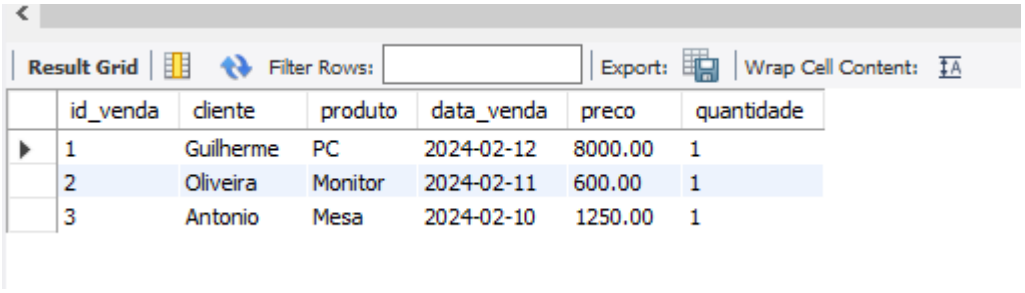
Após parametrizar as configurações do BD, apareceu para conectar




```
CREATE TABLE Vendas(
id_venda INT,
cliente varchar(50),
produto varchar(50),
data_venda date,
preco decimal(6,2),
quantidade int
)
```

```
INSERT INTO Vendas( id_venda, cliente, produto, data_venda, p
INSERT INTO Vendas( id_venda, cliente, produto, data_venda, p
INSERT INTO Vendas( id_venda, cliente, produto, data_venda, p
```

```
SELECT * from Vendas
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| id_venda | cliente | produto | data_venda | preco | quantidade |
|---|---|---|---|---|---|
| 1 | Guilherme | PC | 2024-02-12 | 8000.00 | 1 |
| 2 | Oliveira | Monitor | 2024-02-11 | 600.00 | 1 |
| 3 | Antonio | Mesa | 2024-02-10 | 1250.00 | 1 |

Desta maneira consegui criar o escopo do meu banco de dados, agora é tentar realizar a busca dessas informações pelo Python.

# Python

Realizei pesquisas na internet e consegui buscar a biblioteca para importa a conexão do mysql, aproveitei e coloquei ‘error handle’.

```
import mysql.connector
from mysql.connector import errorcode
try:
    cnx = mysql.connector.connect(user='root', password='exam
except mysql.connector.Error as err:
  if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
    print("Something is wrong with your user name or password
  elif err.errno == errorcode.ER_BAD_DB_ERROR:
    print("Database does not exist")
  else:
    print(err)
else:
  if cnx and cnx.is_connected():

    with cnx.cursor() as cursor:

        result = cursor.execute("SELECT * FROM Vendas LIMIT 5
        rows = cursor.fetchall()
        for rows in rows:
            print(rows)
  cnx.close()
```

Adaptei o código que encontrei para as minhas necessidades e deu certo:

```
Documents/Estudo_data/python_sql/main.pyata\python_sql> & C
(1, 'Guilherme', 'PC', datetime.date(2024, 2, 12), Decimal('8000.00'), 1)
(2, 'Oliveira', 'Monitor', datetime.date(2024, 2, 11), Decimal('600.00'), 1)
(3, 'Antonio', 'Mesa', datetime.date(2024, 2, 10), Decimal('1250.00'), 1)
PS C:\Users\Guilherme\Documents\Estudo_data\python_sql>
```

Mas achei confuso esta maneira de utilizar o try e catch, coloquei no chatGPT para melhorar o código utilizando boas práticas:

> meu código em python faz comunicação com o banco de dados my e faz uma consulta.
> Preciso verificar se está conforme as praticas do código limpo:( colei todo o código)

O retorno do Chat foi muito positivo, pois informou para utilizar o bloco ‘with’ para conexão, utilizar exceções específicas( a boa prática diz que ao invés de capturar todas as exceções, posso lidar de maneira separado)

o código retornado por ele foi o seguinte:

```
import mysql.connector
from mysql.connector import Error, errorcode

try:
    with mysql.connector.connect(user='root', password='examp
        if cnx.is_connected():
            with cnx.cursor() as cursor:
                cursor.execute("SELECT * FROM Vendas LIMIT 5"
                rows = cursor.fetchall()
                for row in rows:
                    print(row)

except mysql.connector.Error as err:
    if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
        print("Erro de acesso: Verifique seu nome de usuário
    elif err.errno == errorcode.ER_BAD_DB_ERROR:
        print("Erro de banco de dados: O banco de dados não e
```

```
    else:
        print(f"Erro inesperado: {err}")
```

A sintaxe ficou mais clara de entender, onde o bloco que faz a comunicação, ja faz a leitura da tabela, enquanto a exceção só lida com os erros.

```
Documents/Estudo_data/python_sql/teste.py
(1, 'Guilherme', 'PC', datetime.date(2024, 2, 12), Decimal('8000.00'), 1)
(2, 'Oliveira', 'Monitor', datetime.date(2024, 2, 11), Decimal('600.00'), 1)
(3, 'Antonio', 'Mesa', datetime.date(2024, 2, 10), Decimal('1250.00'), 1)
PS C:\Users\Guilherme\Documents\Estudo_data\python_sql>
```