**Conectando Python a um banco de dados Mysql**

Universidade Católica de Pelotas

Guilherme Moura Baccarin

**Introdução**

Este projeto tem como objetivo a captação e armazenamento de informações importantes de um computador através da conexão da linguagem de programação Python e o uso de um bancp de dados Mysql. O projeto prevê a leitura de infomações como, percentual de uso de memória RAM, percentual livre do disco, para futuras manipulações dos mesmos, com inúmeras possíves aplicações nos dias atuais.

**Sumário**

[**Introdução** 2](#_Toc52114925)

[**Bibliotecas** 5](#_Toc52114926)

[ SQLAlchemy 5](#_Toc52114927)

[ Platform 6](#_Toc52114928)

[ Os 6](#_Toc52114929)

[ Time 7](#_Toc52114930)

[ Psutil 7](#_Toc52114931)

[ Mysql.connector 7](#_Toc52114932)

[**Dados obtidos** 9](#_Toc52114933)

[**Implementação** 10](#_Toc52114934)

[**Estrutura do banco** 13](#_Toc52114935)

[**Consultas no banco** 14](#_Toc52114936)

[**Conclusão** 15](#_Toc52114937)

**Bibliotecas**

Para o desenvolvimento de algoritmos mais complexos em quase qualquer linguagem de programação é necessário o uso de bibliotecas.

Biblioteca é uma coleção de [subprogramas](https://pt.wikipedia.org/wiki/Subprograma) utilizados no desenvolvimento de [software](https://pt.wikipedia.org/wiki/Software). Bibliotecas contém código e dados auxiliares, que provém serviços a programas independentes, o que permite o compartilhamento e a alteração de código e dados de forma [modular](https://pt.wikipedia.org/wiki/M%C3%B3dulo_(programa%C3%A7%C3%A3o)).

* SQLAlchemy-SQLAlchemy é um [**framework**](https://www.treinaweb.com.br/blog/para-que-serve-um-framework/) de mapeamento objeto-relacional SQL (ORM) , [**ORM**](https://www.treinaweb.com.br/blog/o-que-e-orm/) trata-se de um Framework que visa auxiliar na redução da impedância, realizando todas as conversões necessárias entre o modelo relacional e o modelo orientado a objetos de maneira automática.

1. Session = sessionmaker(bind=engine)
2. session = Session()
3. f1 = Filme("Star Trek", 2009)
4. f1.diretor = Diretor("JJ Abrams")
5. d2 = Diretor("George Lucas")
6. d2.filmes = [Filme("Guerra nas Estrelas", 1977), Filme("THX 1138", 1971)]
7. **try**:
8. session.add(f1)
9. session.add(d2)
10. session.commit()
11. **except**:
12. session.rollback()

* Platform- Usado para acessar os dados da plataforma subjacente, como hardware, sistema operacional e informações de versão do interpretador.

1. Import platform
2. #Imprime versão do python instalada
3. Print(‘Versão do Python :’ , platform.python\_version())
4. #Imprime a arquitetura do computador
5. print('Arquitetura:', platform.architecture())
6. #Imprime o sistema operacional instalado
7. print('Sistema operacional :', platform.system())

* Os– Um dos principais módulos em Python, traz informações sobre o sistema operacional.

1. **import** **os**
2. #Verifica se existe o diretório
3. **if** os.path.exists("z"):
4. **print**("O diretório z existe.")
5. **else**: **print**("O diretório z não existe.")
6. #Imprime o diretório atual
7. os.getcwd()

* Time **(Utilizada)** - Usada para possibilitar um intervalo entre as leituras.

1. import time
2. print("This is printed immediately.")
3. time.sleep(2.4) #Determina uma pausa na execução
4. print("This is printed after 2.4 seconds.")
5. named\_tuple = time.localtime() # get struct\_time
6. time\_string = time.strftime("%m/%d/%Y, %H:%M:%S", named\_tuple) #Formata a data em determinado molde
7. print(time\_string)

* Psutil **(Utilizada)** – Usada para recuperar informações sobre processos em execução e utilização sistema.

1. import psutil
3. #Imprime o número de cores no sistema
4. print("Number of cores in system", psutil.cpu\_count())
6. #Imprime informações sobre o swap na máquina
7. print(psutil.swap\_memory())
8. print(psutil.disk\_usage('/'))
9. #Imprime informações sobre o uso do disco “/”

* Mysql.connector **(Utilizada)** – Permite que programas Python acessem bancos de dados MySQL.

1. import mysql.connector
2. mydb = mysql.connector.connect(
3. host="localhost",
4. user="yourusername",
5. password="yourpassword"
6. )
7. print(mydb)

**Dados obtidos**

A escolha de quais dados que serão salvos é de grande importância, pois cada dado é relevante pra certas coisas, e armazenar um dado que não é importante para a sua análise só deixará a sua base mais pesada, poluida e mais confusa.

Então é importante dedicar um tempo para a anáise de quais dados são relevantes para a sua análise.

* **Valor usado de disco** – Importante manter um controle sob a utilizalção do disco, caso aconteça uma invasão de um usuário não desejado e esteja usando a máquina para fins não desejados.
* **Valor de disco livre** - Importante manter um controle sob a utilizalção do disco para eventuais necessidades de um upgrade de máquina.
* **Valor de disco livre percentual** – Valor mais visível da utilização do disco.
* **Valor de swap percentual** – Importante se ter conhecimento da quantidade de swap que a máquina está utilizando, pois caso esteja sendo usado um valor alto, deixará a máquina mais lenta.
* **Valor de memória RAM livre** – Um valor muito usado para medir a velocidade de processamento da máquina.
* **Valor de memória RAM livre percentual** – Uma forma mais palpável de visualizar o uso de memória RAM.
* **Data de leitura** – Indispensável saber quando foi feita a leitura das informações.

**Implementação**

* **Conexão com o banco de dados.**

Parte onde é feita a conexão do python com o banco de dados MySQL, e feita a verificação se houve algum erro nesse processo.

1. connection = mysql.connector.connect(
2. #Informações da base de dados
3. host='localhost', database='BASE', user='root',
4. password='SENHA')
5. if connection.is\_connected(): #Caso funcione a conexão
6. db\_Info = connection.get\_server\_info() #Info do db
7. print("Conectado a MySQL Server versão ", db\_Info)
8. cursor = connection.cursor()
9. cursor.execute("select database();") #Retorna nome da base
10. record = cursor.fetchone()
11. print("Base conectada: ", record)

* **Informa qual disco**

Informado qual o disco que se deve ser lido para o restante do código.

1. discoG = psutil.disk\_usage('G://')
2. discoC = psutil.disk\_usage('C://')
3. discoE = psutil.disk\_usage('E://')

* **Swap**

1. swap = psutil.swap\_memory()

* **RAM**

1. ram = psutil.virtual\_memory()

* **Insert**

1. #Texto base do insert na base de dados
2. insert = """insert into leitura (id\_disco,valor\_usado,valor\_livre,valor\_livre\_percentual,
3. swap\_percentual,ram\_livre,ram\_livre\_percentual)
4. values (%s,%s,%s,%s,%s,%s,%s);"""

* **Leituras**

Executada as leituras para cada disco para os valores de uso de disco, valor livre restante do disco e o valor em percentual livre do mesmo.

* + **Disco G**

1. valor\_usadoG = round(discoG.used / (1024.0 \*\* 3), 2)
2. valor\_livreG = round(discoG.free / (1024.0 \*\* 3), 2)
3. valor\_livre\_percentualG = discoG.percent
   * **Disco C**
4. valor\_usadoC = round(discoC.used / (1024.0 \*\* 3), 2)
5. valor\_livreC = round(discoC.free / (1024.0 \*\* 3), 2)
6. valor\_livre\_percentualC = discoC.percent
   * **Disco E**
7. valor\_usadoE = round(discoE.used / (1024.0 \*\* 3), 2)
8. valor\_livreE = round(discoE.free / (1024.0 \*\* 3), 2)
9. valor\_livre\_percentualE = discoE.percent
   * **Swap**
10. swap\_percentual = round(swap.percent, 2)
    * **RAM**
11. ram\_livre = round(ram.percent, 2)
12. ram\_livre\_percentual = round(ram.available \* 100 / ram.total, 2)

* **Executa**

1. #Executando os inserts na base de dados
2. cursor.execute(insert, (id\_disco\_G,valor\_usadoG,valor\_livreG,valor\_livre\_percentualG,swap\_percentual,ram\_livre,ram\_livre\_percentual))
3. cursor.execute(insert, (id\_disco\_C,valor\_usadoC,valor\_livreC,valor\_livre\_percentualC,swap\_percentual,ram\_livre,ram\_livre\_percentual))
4. cursor.execute(insert, (id\_disco\_E,valor\_usadoE,valor\_livreE,valor\_livre\_percentualE,swap\_percentual,ram\_livre,ram\_livre\_percentual))

* **Commit**

1. #Envia o insert para o banco de dados
2. connection.commit()

* **Sleep**

Tempo entre as leituras.

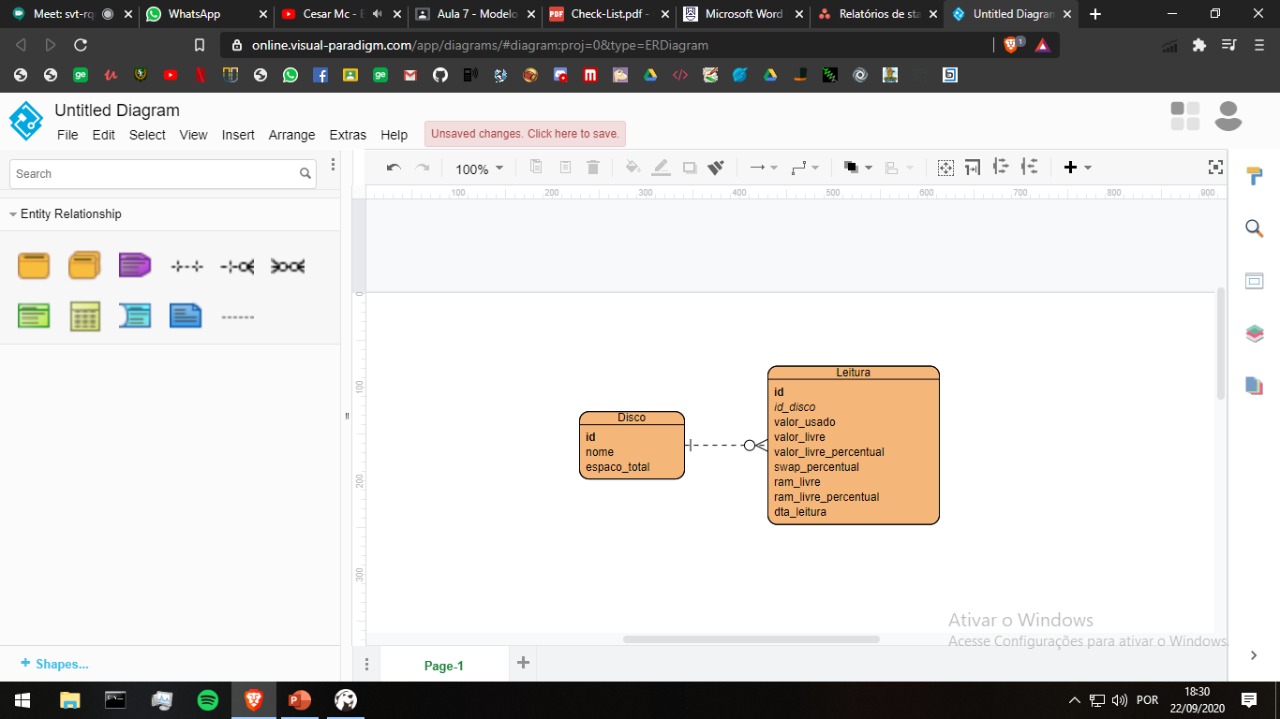
1. time.sleep(50)

**Estrutura do banco**

* + **SQL**

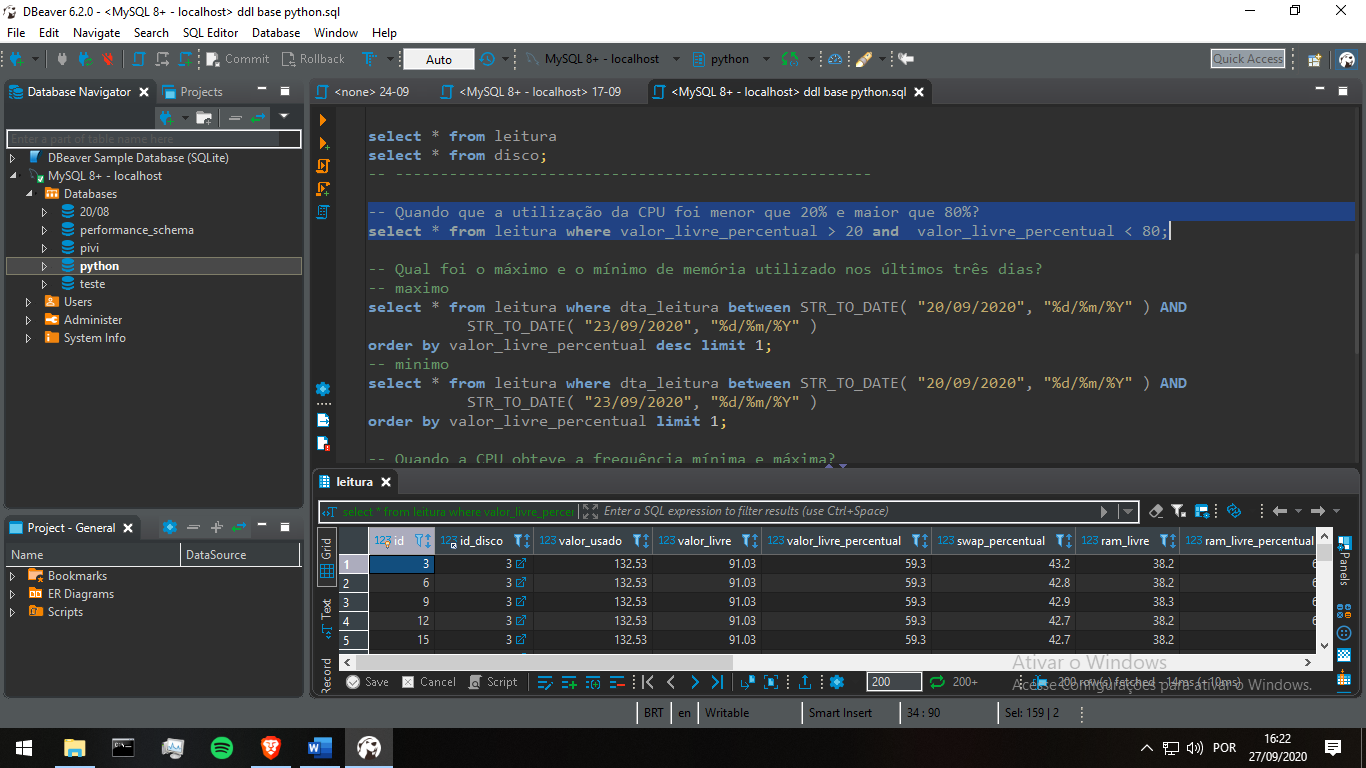
1. **create** **table** disco (
2. id **int** **unique** auto\_increment **primary** **key**,
3. nome **varchar**(100) **not** **null**,
4. espaco\_total **float** **not** **null**
5. );
6. **create** **table** leitura(
7. id **int** **unique** auto\_increment **primary** **key**,
8. id\_disco **int** **not** **null**,
9. valor\_usado **float** **not** **null**,
10. valor\_livre **float** **not** **null**,
11. valor\_livre\_percentual **float** **not** **null**,
12. swap\_percentual **float** **not** **null**,
13. ram\_livre **float** **not** **null**,
14. ram\_livre\_percentual **float** **not** **null**,
15. dta\_leitura **TIMESTAMP** **NOT** **NULL** **DEFAULT** **CURRENT\_TIMESTAMP**,
16. **FOREIGN** **KEY** (id\_disco) **REFERENCES** disco(id)
17. );
    * **Entidade-Relacionamento**

Um diagrama entidade relacionamento (ER) é um tipo de fluxograma que ilustra como “entidades”, p. ex., pessoas, objetos ou conceitos, se relacionam entre si dentro de um sistema. Diagramas ER são mais utilizados para projetar ou depurar bancos de dados relacionais nas áreas de engenharia de software, sistemas de informações empresariais, educação e pesquisa. Também conhecidos como DERs, ou modelos ER, usam um conjunto definido de símbolos, tais como retângulos, diamantes, ovais e linhas de conexão para representar a interconectividade de entidades, relacionamentos e seus atributos.

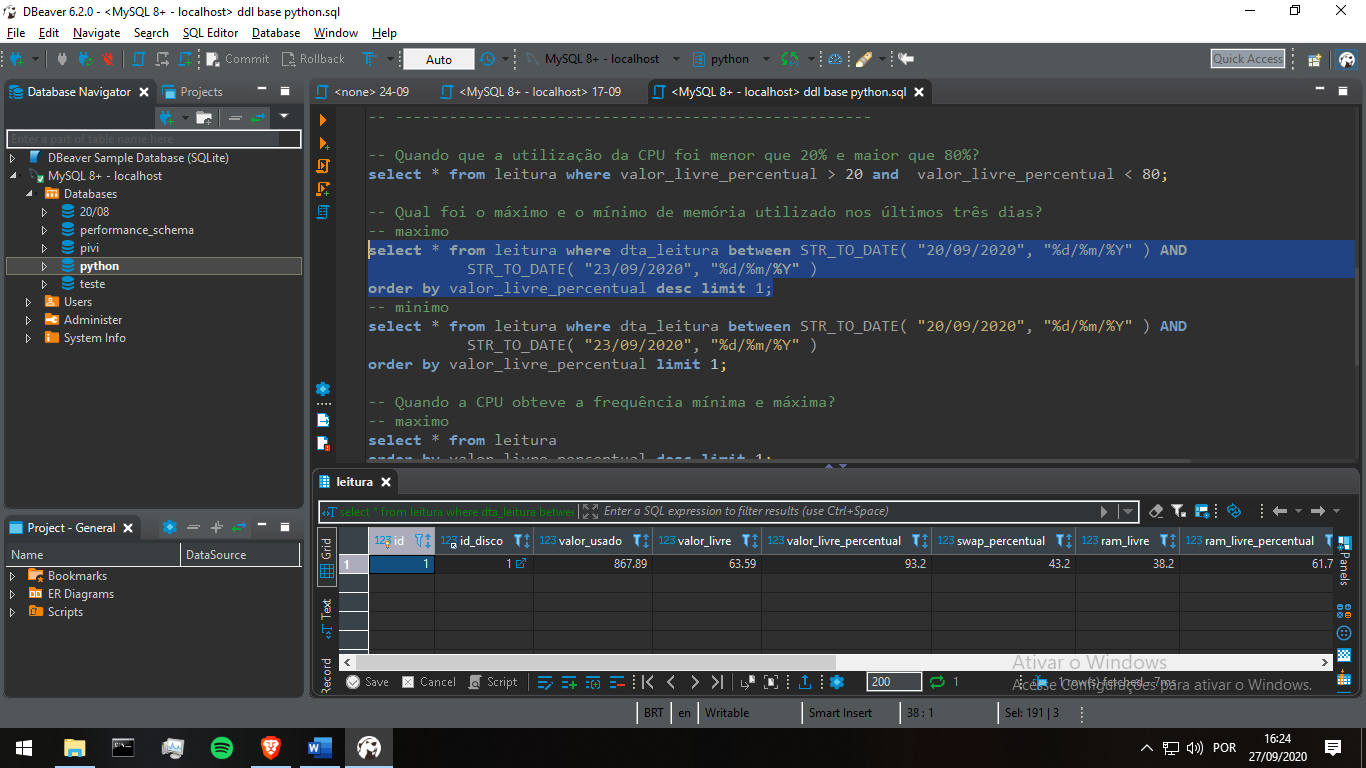


**Consultas no banco**

* + **-- Quando que a utilização da CPU foi menor que 20% e maior que 80%?**
    1. **select** \* **from** leitura **where** valor\_livre\_percentual > 20 **and** valor\_livre\_percentual < 80;



* + **-- Qual foi o máximo e o mínimo de memória utilizado nos últimos três dias?**
    - * **máximo**
      1. **select** \* **from** leitura **where** dta\_leitura **between** STR\_TO\_DATE( "20/09/2020", "%d/%m/%Y" ) **AND**
      2. STR\_TO\_DATE( "23/09/2020", "%d/%m/%Y" )
      3. **order** **by** valor\_livre\_percentual **desc** **limit** 1;

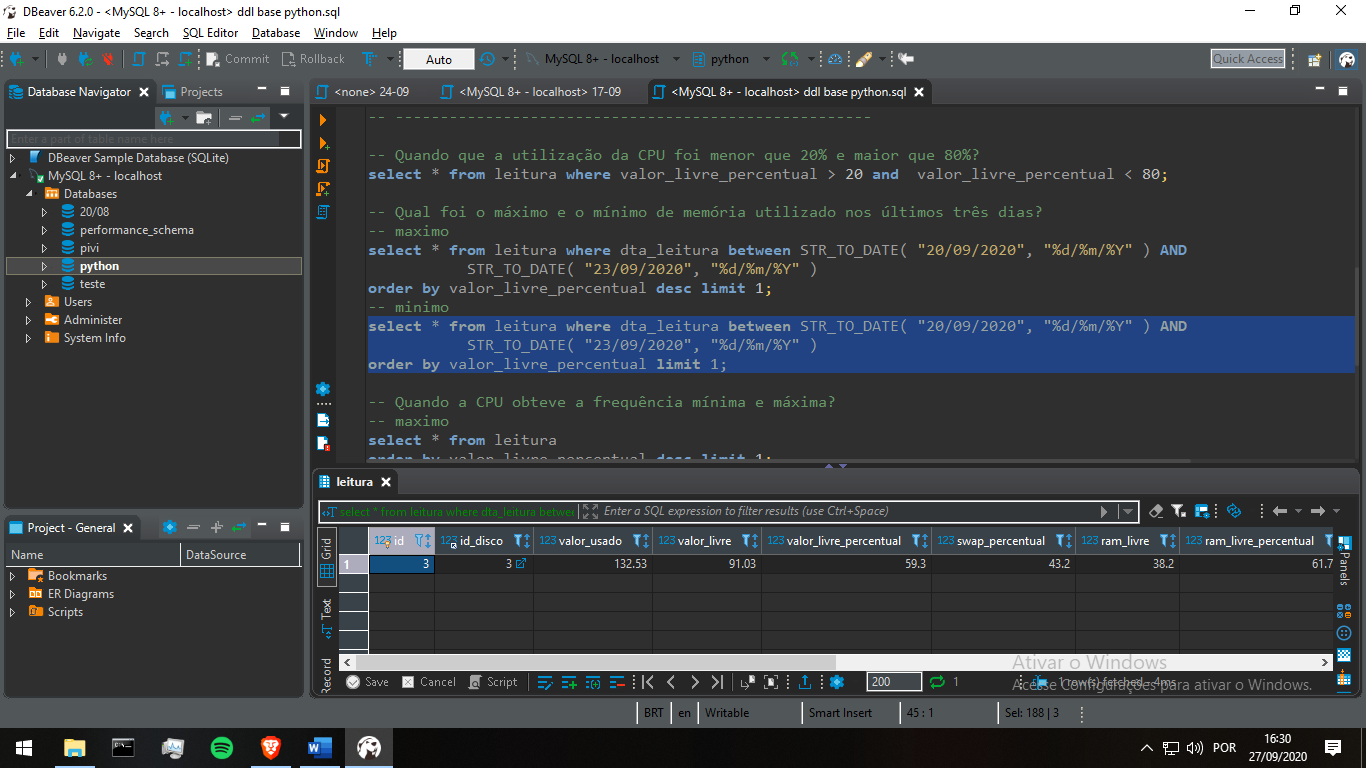


* + - 1. **minimo**

1. **select** \* **from** leitura **where** dta\_leitura **between** STR\_TO\_DATE( "20/09/2020", "%d/%m/%Y" ) **AND**

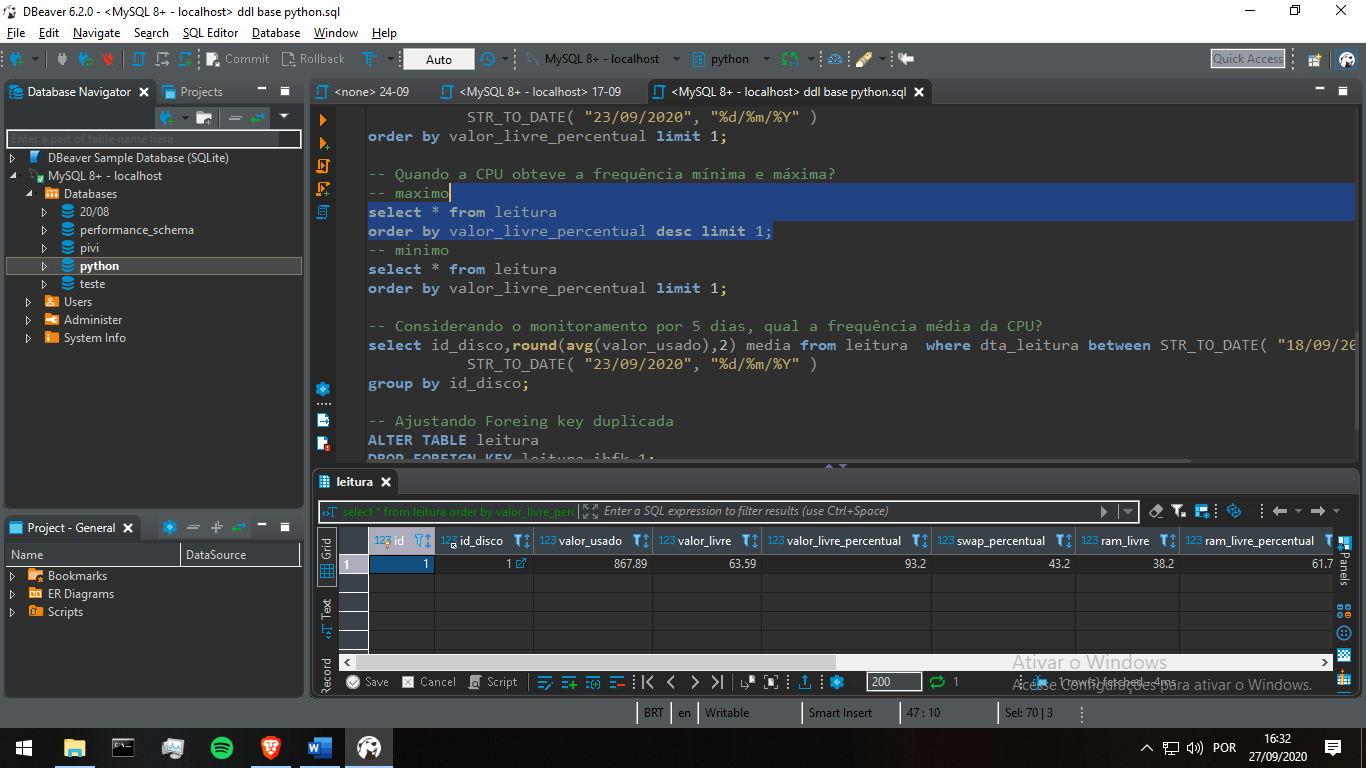
STR\_TO\_DATE( "23/09/2020", "%d/%m/%Y" )

1. **order** **by** valor\_livre\_percentual **limit** 1;



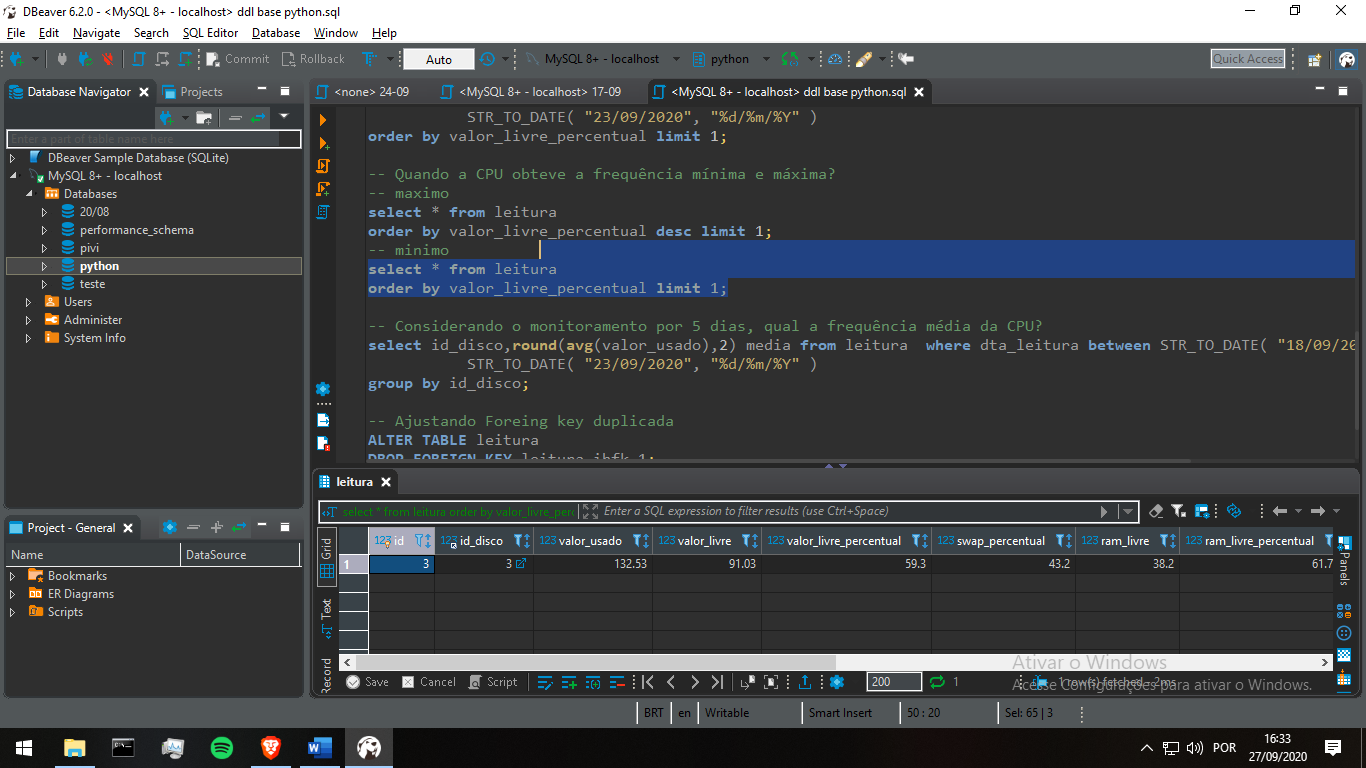
* + **-- Quando a CPU obteve a frequência mínima e máxima?**
    - * **-- maximo**

1. **select** \* **from** leitura
2. **order** **by** valor\_livre\_percentual **desc** **limit** 1;



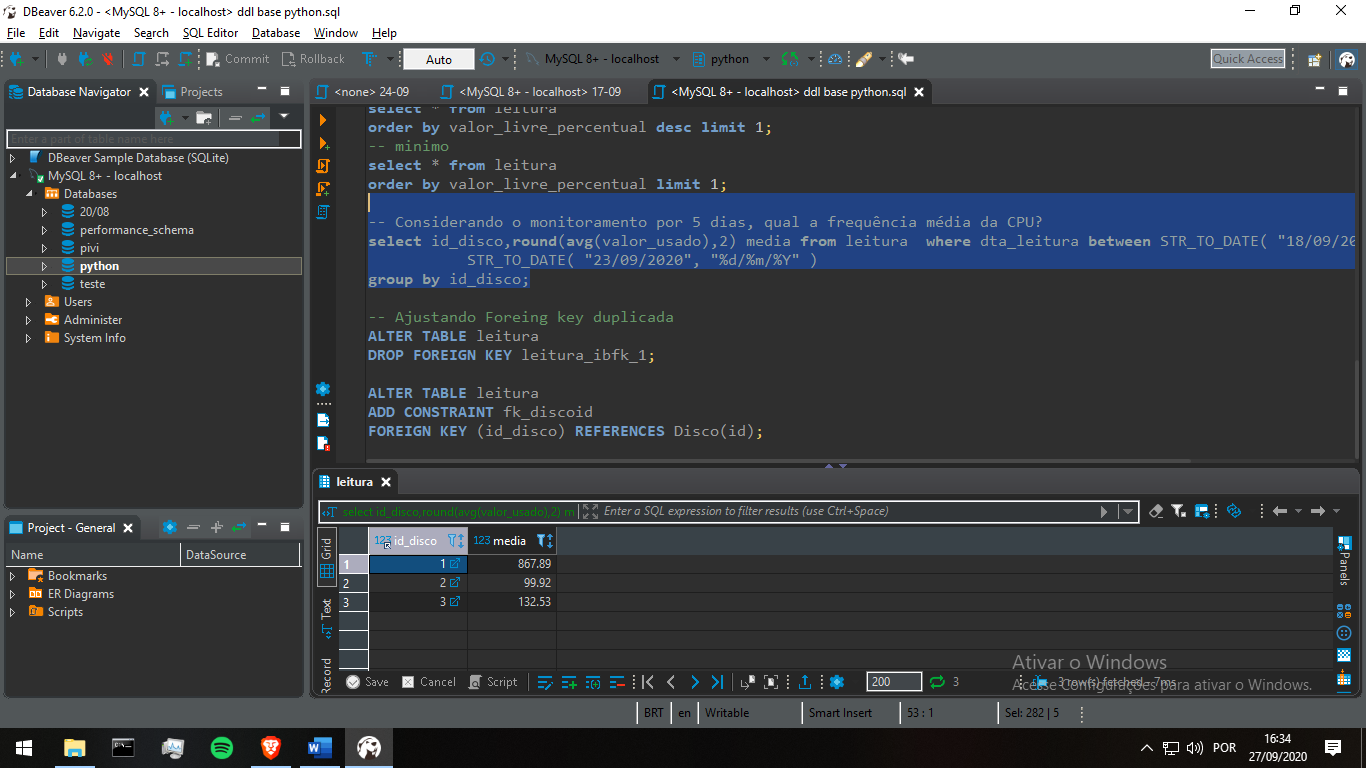
**-- minimo**

1. **select** \* **from** leitura
2. **order** **by** valor\_livre\_percentual **limit** 1;



* + **-- Considerando o monitoramento por 5 dias, qual a frequência média da CPU?**

1. **select** id\_disco,round(**avg**(valor\_usado),2) media
2. **from** leitura **where** dta\_leitura **between**
3. STR\_TO\_DATE( "18/09/2020", "%d/%m/%Y" ) **AND**
4. STR\_TO\_DATE( "23/09/2020", "%d/%m/%Y" )
5. **group** **by** id\_disco;



**Conclusão**

Por meio desse projeto, foi possível visualizar que a implementação de uma conexão da linguagem de programação python e um banco de dados mysql é uma tarefa simples mas muito poderosa, pois a usabilidade da mesma é enorme.