Professor: Ruan Chaves Machado

Data do Material Didático Criado: 31/05/2025

Data da Aula: 03/06/2025

Tópicos e Temas abordados nessa aula:

* Tipos de Variáveis Mais Usadas em Integração com Banco de Dados
* Introdução à Linguagem SQL
* Criando um Banco de Dados no MySQL: Instalação, Criação e Conexão via Python

**Tipos de Variáveis Mais Usadas em Integração com Banco de Dados**

**O que são tipos de variáveis nesse contexto?**

Quando desenvolvemos aplicações em Python que interagem com bancos de dados relacionais (como MySQL, PostgreSQL, etc.), precisamos entender **quais tipos de dados usamos em Python** e **como eles são armazenados ou mapeados** em SQL.

Por exemplo: uma string no Python vai ser gravada como VARCHAR ou TEXT no banco.

**Mapeamento entre tipos de dados Python e SQL**

| **Tipo em Python** | **Equivalente em SQL** | **Observações** |
| --- | --- | --- |
| int | INT, BIGINT | Para valores numéricos inteiros |
| float | FLOAT, DOUBLE | Para valores numéricos com casas decimais |
| str | VARCHAR, TEXT | Para textos curtos ou longos |
| bool | BOOLEAN, TINYINT(1) | Armazenado como 1 (True) ou 0 (False) no MySQL |
| datetime.datetime | DATETIME, TIMESTAMP | Requer importação do módulo datetime |
| date | DATE | Apenas a data, sem horário |
| list / dict | JSON | Suportado em MySQL 5.7+, útil para armazenar dados estruturados dinamicamente |

**Exemplo prático de uso desses dados em Python**

**from** datetime import datetime

# Suponha que vamos inserir esses dados **no** banco**:**

aluno **=** **{**

"nome"**:** "João da Silva"**,**

"idade"**:** 23**,**

"ativo"**:** **True,**

"media"**:** 8.5**,**

"data\_cadastro"**:** datetime**.**now**()**

**}**

Quando for preparar a query SQL, você faz o binding desses dados com os placeholders:

**sql** **=** "INSERT INTO alunos (nome, idade, ativo, media, data\_cadastro) VALUES (%s, %s, %s, %s, %s)"

valores **=** **(**aluno**[**"nome"**],** aluno**[**"idade"**],** aluno**[**"ativo"**],** aluno**[**"media"**],** aluno**[**"data\_cadastro"**])**

cursor**.execute(sql,** valores**)**

‘

É possível notar que que:

* str → VARCHAR
* int → INT
* bool → BOOLEAN (TINYINT(1) no MySQL)
* float → FLOAT
* datetime → DATETIME

Para memorizar + Analogia:

| **Tipo Python** | **Analogias / Comparações Simples** |
| --- | --- |
| int | Contagem de moedas, idade |
| float | Notas de prova, saldo bancário |
| str | Nome, endereço, qualquer texto |
| bool | Interruptor ligado/desligado (True/False) |
| datetime | Registro de entrada de um funcionário |
| dict / list | Estrutura como carrinho de compras (dados flexíveis) |

**O que é SQL?**

SQL (Structured Query Language) é uma linguagem **estruturada** e **declarativa** usada para interagir com bancos de dados relacionais. Ela permite que você:

* **Crie e modifique** estruturas de banco de dados (tabelas, índices, etc.);
* **Insira, atualize, remova** dados;
* **Consulte** dados com critérios e ordenações;
* **Controle o acesso** aos dados (permissões, segurança).

Você **declara o que quer** (ex: "me dê todos os alunos com nota acima de 8") e o banco se encarrega de **como fazer isso internamente**.

**Principais comandos SQL**

| **Comando** | **Função Principal** | **Exemplo** |
| --- | --- | --- |
| CREATE TABLE | Criar uma nova tabela | CREATE TABLE alunos (...) |
| INSERT INTO | Inserir novos dados | INSERT INTO alunos VALUES (...) |
| SELECT | Consultar dados | SELECT \* FROM alunos |
| UPDATE | Atualizar dados existentes | UPDATE alunos SET nome = 'Carlos' |
| DELETE | Remover dados | DELETE FROM alunos WHERE id = 1 |
| DROP TABLE | Apagar uma tabela do banco | DROP TABLE alunos |
| ALTER TABLE | Alterar a estrutura de uma tabela | ALTER TABLE alunos ADD COLUMN idade |
| WHERE | Condição de filtragem | SELECT \* FROM alunos WHERE nota > 7 |
| ORDER BY | Ordenação de dados | SELECT \* FROM alunos ORDER BY nota |
| JOIN | Unir dados de múltiplas tabelas relacionadas | SELECT \* FROM alunos JOIN cursos ... |

**Exemplo simples de uso**

-- Criando a tabela

**CREATE** **TABLE** alunos **(**

id INT AUTO\_INCREMENT **PRIMARY** **KEY,**

nome VARCHAR**(**100**),**

nota FLOAT

**);**

-- Inserindo dados

**INSERT** **INTO** alunos **(**nome**,** nota**)** **VALUES** **(**'João'**,** 8.5**),** **(**'Maria'**,** 9.2**);**

-- Consultando os dados

**SELECT** **\*** **FROM** alunos**;**

-- Atualizando um aluno

**UPDATE** alunos **SET** nota **=** 9.5 **WHERE** nome **=** 'João'**;**

-- Removendo um aluno

**DELETE** **FROM** alunos **WHERE** nome **=** 'Maria'**;**

Para memorizar (exemplo como se fosse um Excel):

* **Tabelas** = Planilhas do Excel (linhas e colunas);
* **Cada** **linha** = um registro (um aluno, um produto, etc.);
* **SQL** = comandos que você dá para o Excel filtrar, editar ou calcular as planilhas.

**Criando um Banco de Dados no MySQL: Instalação, Criação e Conexão via Python**

O **MySQL** é um dos SGBDs (Sistemas de Gerenciamento de Banco de Dados) mais usados no mundo. É gratuito, open source e funciona muito bem em conjunto com linguagens como Python, PHP e Java.

* **MySQL é relacional** → trabalha com tabelas, relacionamentos e dados estruturados.
* **Multiusuário e multitarefa** → vários usuários podem acessá-lo ao mesmo tempo.
* Muito usado em aplicações web e sistemas comerciais.

### Como Baixar e Instalar o MySQL

#### Site oficial (banco):

<https://dev.mysql.com/downloads/installer/>

Site oficial (SGBD – Workbench)

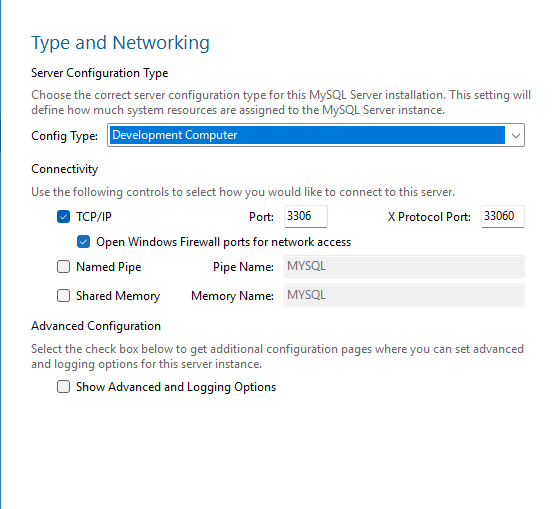
[https://dev.mysql.com/downloads/workbench/](https://dev.mysql.com/downloads/workbench/" \t "_new)

#### **O que instalar:**

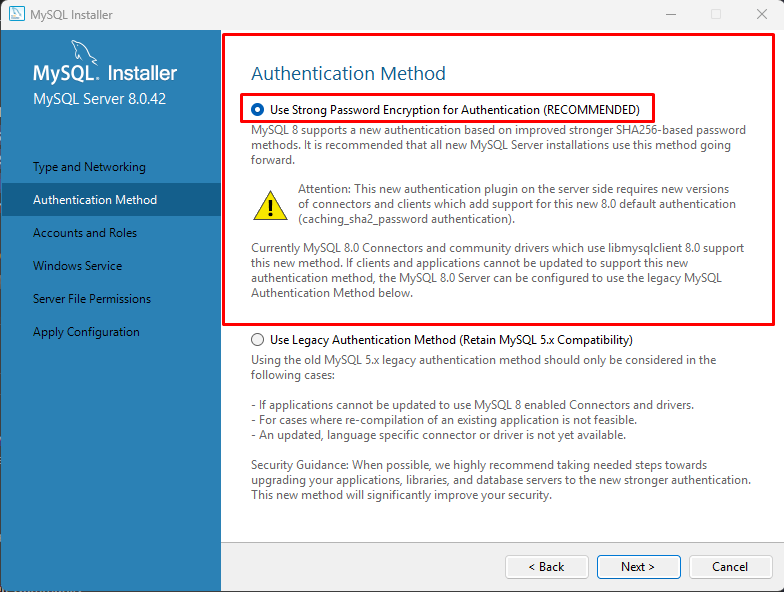
Durante a instalação com o MySQL Installer, escolha instalar os seguintes componentes:

* MySQL Server Only
* MySQL Workbench (ferramenta visual para gerenciar o banco – MySQL Server)

**1º Type and Networking: Config Type: Development Computer (Imagem abaixo)**

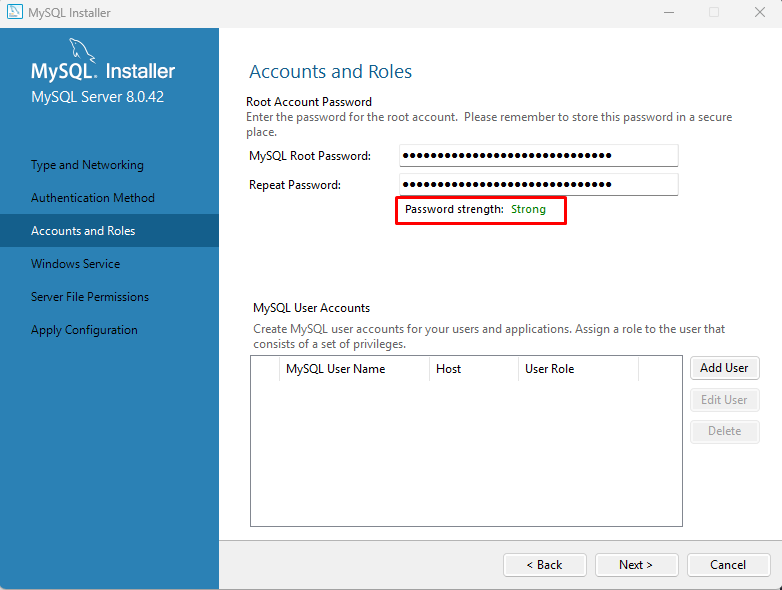
**

**2º Authentication Method (Imagem abaixo):**



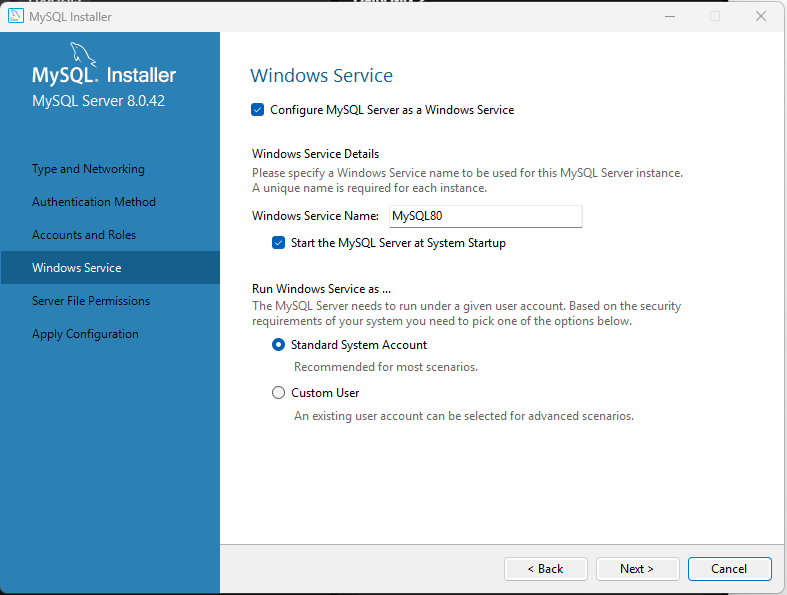
**3º Accounts and Roles (Imagem Abaixo)**

**SALVAR SUA SENHA MUITO BEM! - ici4ZLaBp7eq%uEY@^QVdDJ4G839Va**

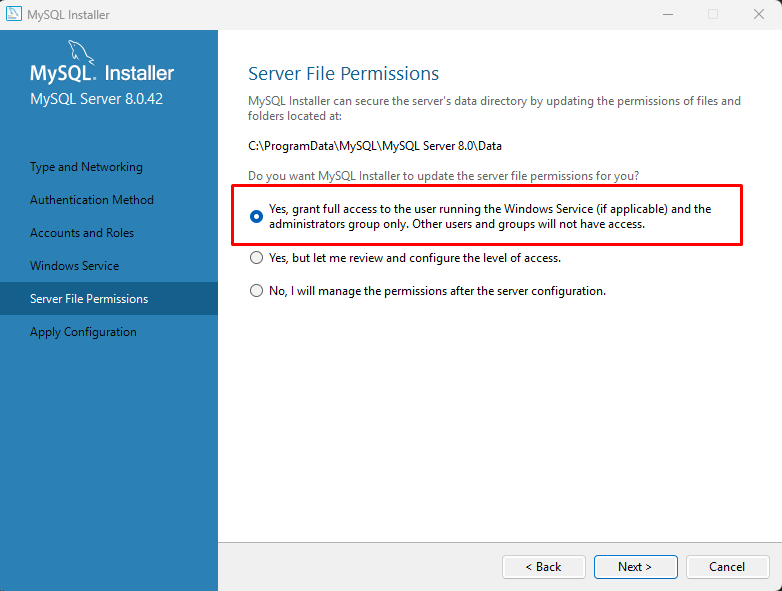


**Site para gerar senhas:** <https://www.roboform.com/br/password-generator>

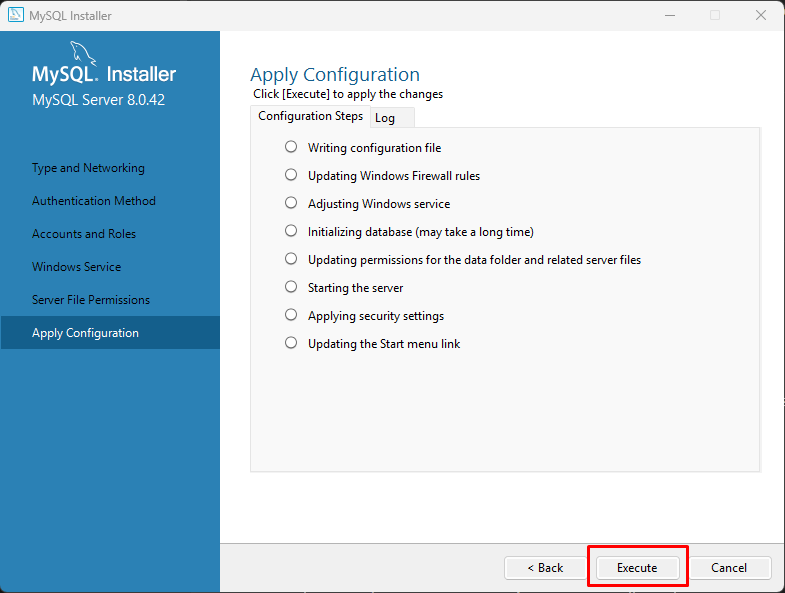
**4º Windows Service (Imagem Abaixo)**

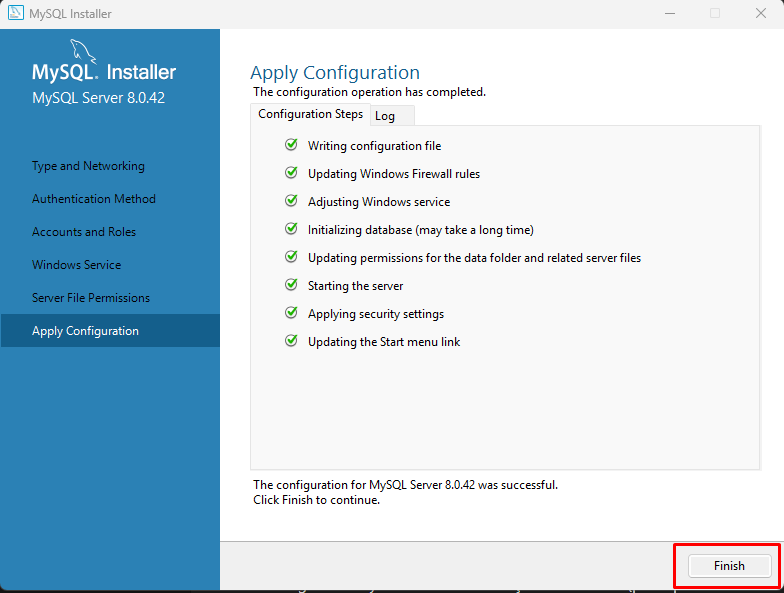


**5º Server File Permissions (Imagem Abaixo)**



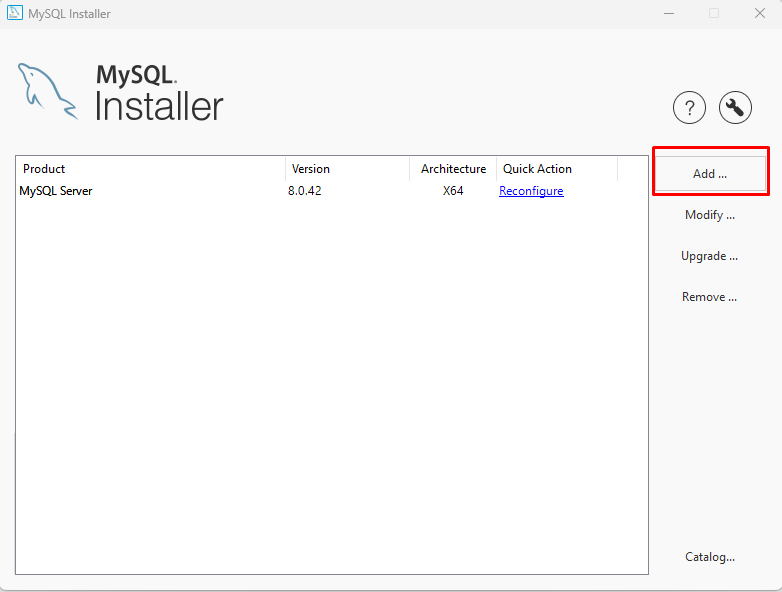
**6º Apply Configuration (Imagem Abaixo – Último passo de Configuração)**



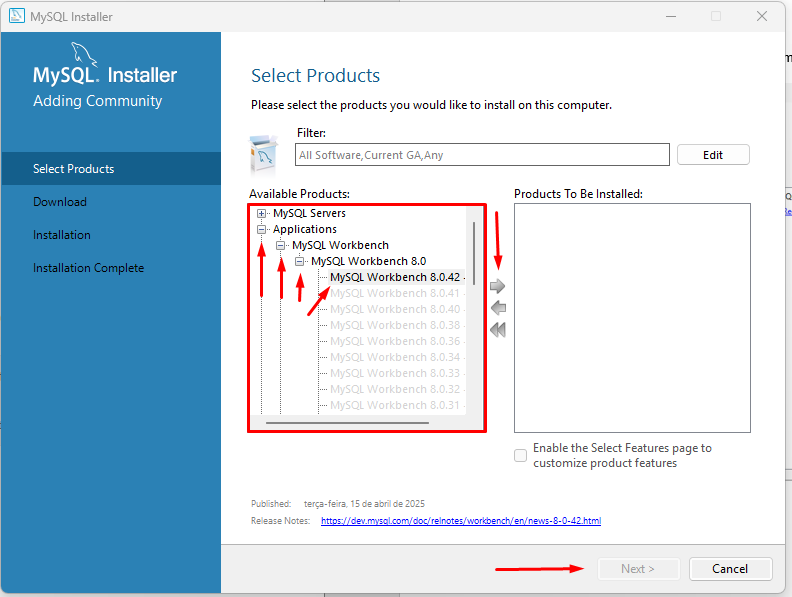


**1.1º Download do SGBD (Workbench)**

Após baixar o Banco MySQL, abra MySQL Installer – Community e siga os seguintes passos:

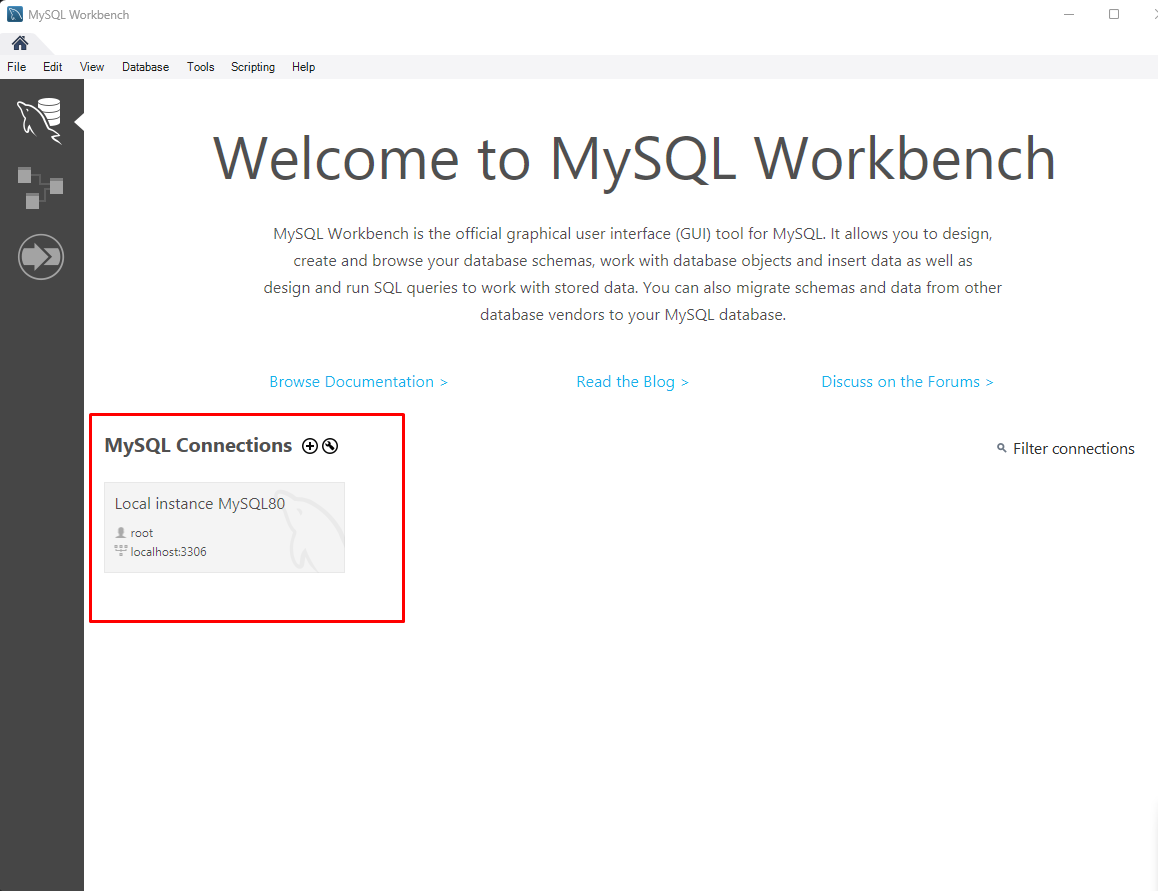


**1.2º Siga os seguintes passos para realizar o Download:**



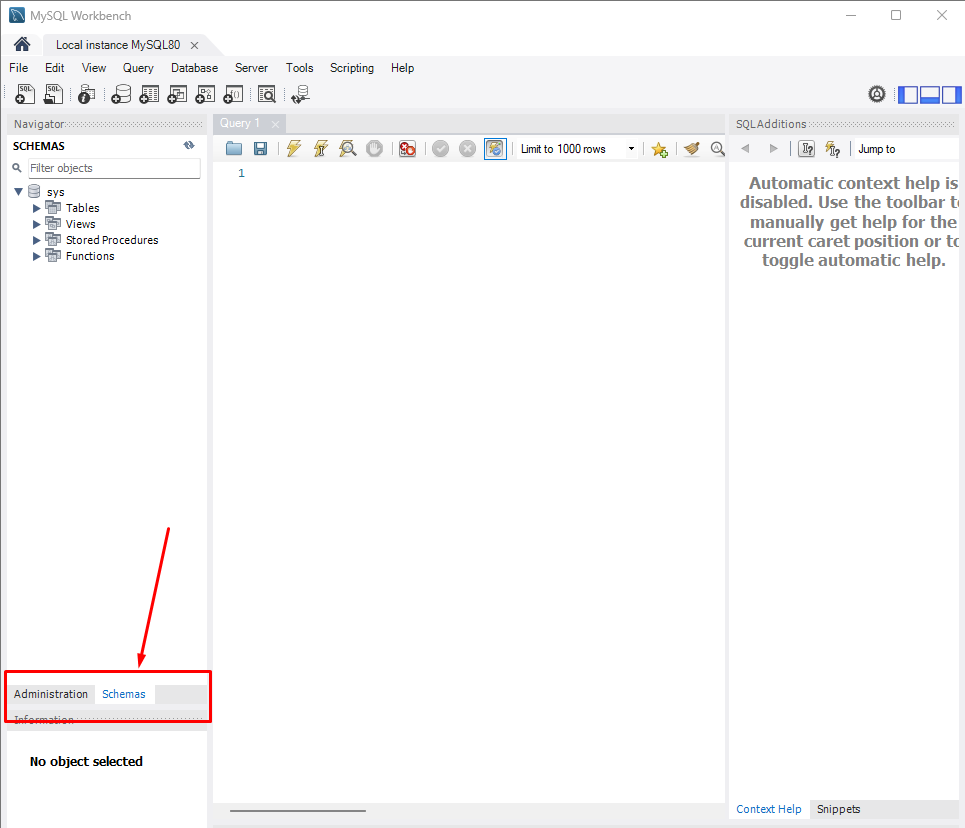
**1.3 Finalize a instalação e Abra o Programa!**

**A sua instância (banco) já estará visível e disponível para uso.**

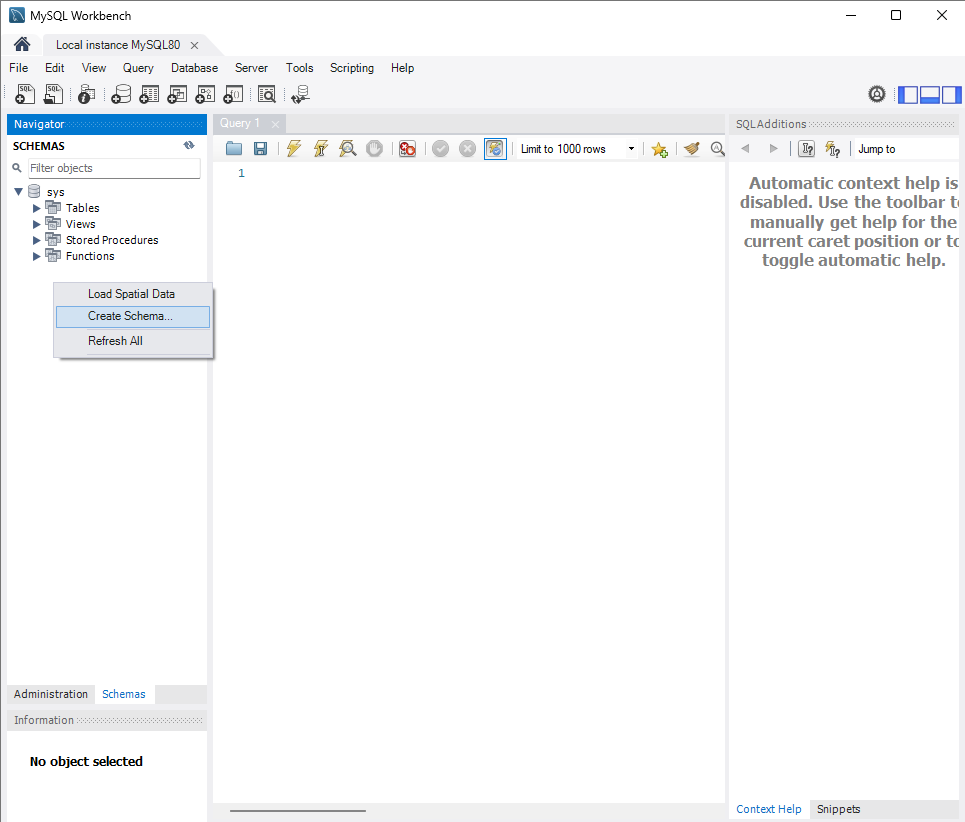


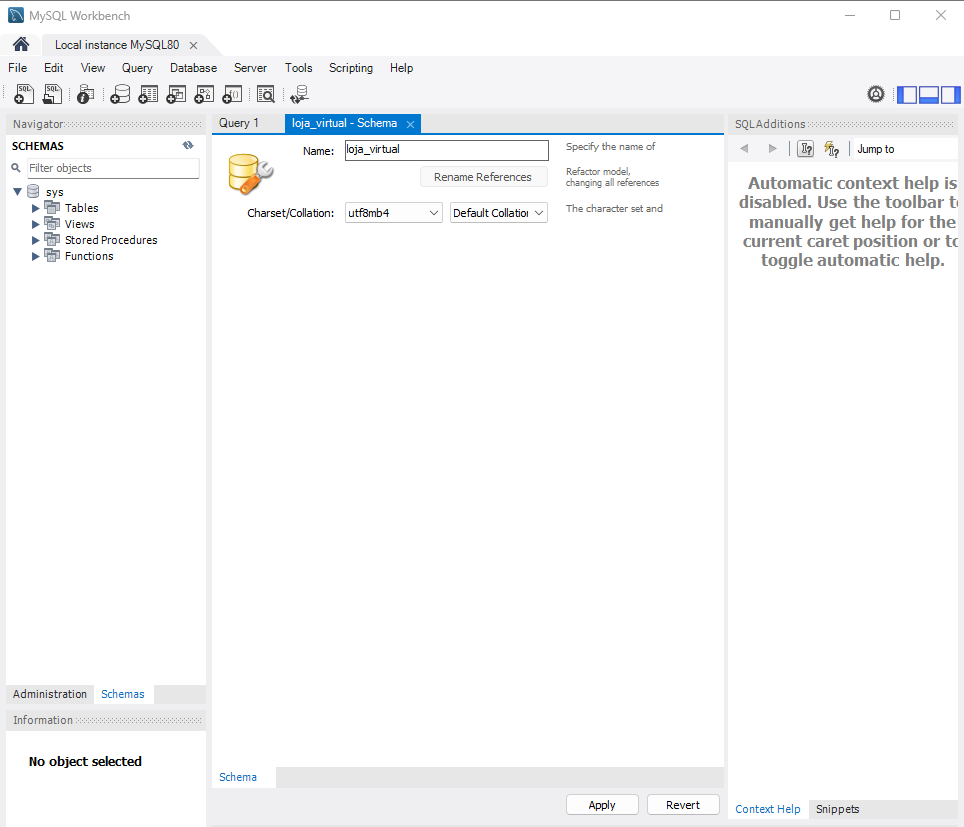
**2.0 Criando uma tabela pela Workbench**

Na aba lateral esquerda, selecione Schemas (imagem abaixo):



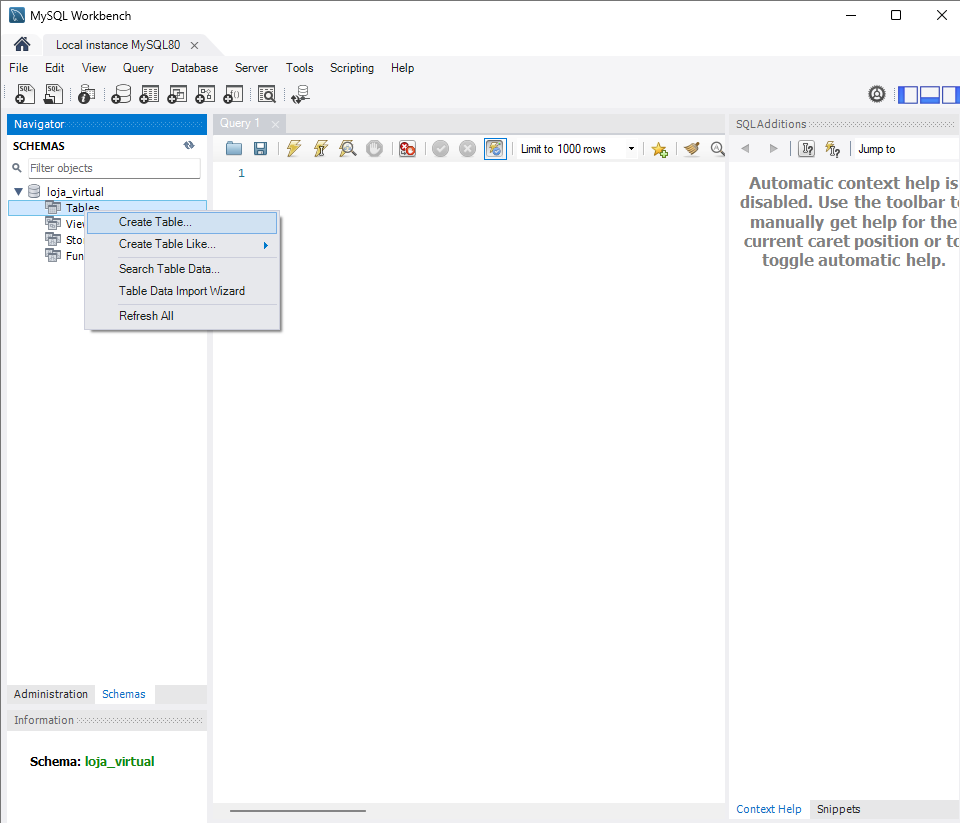
**2.1 Criando um novo Schema (imagem abaixo)**Pressione o botão direito do mouse, e clique em ‘Create Schema...’ e use as configurações abaixo:

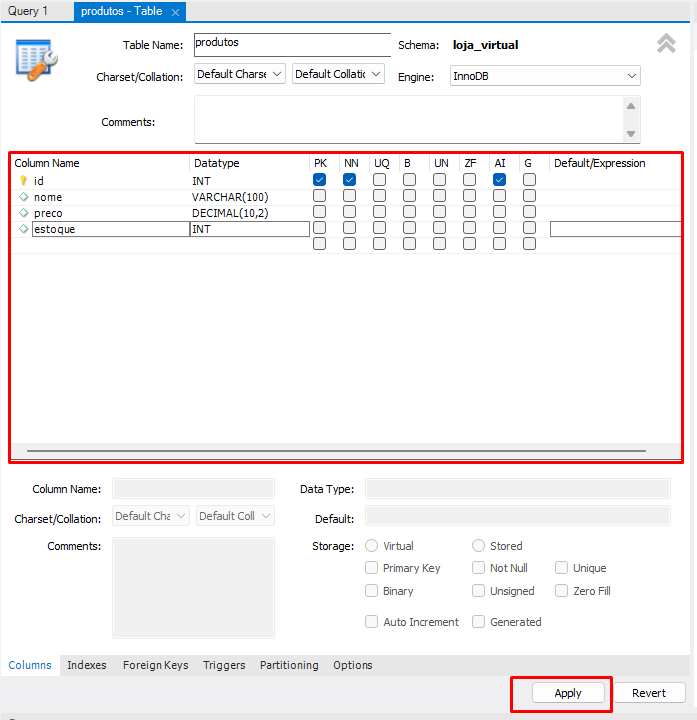
****



**2.2 Criando uma Tabela (produtos)**

Conforme a imagem abaixo, abra ‘loja\_virtual’, clique com o botão direito em ‘Tables’ e depois em ‘Create Table...’ definindo seu nome como ‘produtos’.

****



Marque PK (Primary Key) para id; marque NN (Not Null) para todas, e AI (Auto Increment) para id.

Clique **Apply → Apply** para confirmar; o Workbench mostrará o SQL equivalente:

CREATE TABLE **`**produtos**`** **(**

**`id`** INT NOT NULL AUTO\_INCREMENT**,**

**`**nome**`** VARCHAR**(**100**)** NOT NULL**,**

**`**preco**`** DECIMAL**(**10**,**2**)** NOT NULL**,**

**`**estoque**`** INT NOT NULL**,**

PRIMARY KEY **(`id`)**

**);**

**2.3 Inserindo Valores na Tabela:**

Para a próxima instrução, vamos realizar a seguinte Query:

USE loja\_virtual**;**

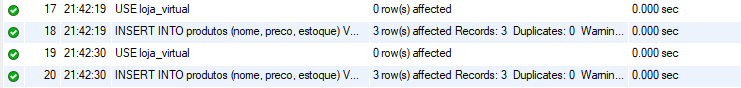
INSERT INTO produtos **(**nome**,** preco**,** estoque**)** VALUES

**(**'Mouse Gamer'**,** 149.90**,** 50**),**

**(**'Teclado Mecânico'**,** 299.90**,** 30**),**

**(**'Monitor 27"'**,** 899.00**,** 15**);**

Isso deve gerar um log parecido com isso:



**2.4 Verificando os Dados:**

Apague essa query atual, e digite:

SELECT **\*** FROM produtos**;**

**Conectando Banco de Dados com PyMySQL**

## **O que é PyMySQL?**

**PyMySQL** é uma biblioteca **100% escrita em Python** que permite a conexão e manipulação de bancos de dados **MySQL** ou **MariaDB** sem depender de bibliotecas compiladas (como mysqlclient, que precisa de compilação em C).

Ela implementa o protocolo do MySQL em Python puro, o que facilita a instalação, portabilidade e uso, especialmente em ambientes como Windows e servidores que não possuem bibliotecas nativas do MySQL.

## **Instalação do PyMySQL**

Você pode instalar o PyMySQL com o pip:

pip install pymysql

**Como conectar ao MySQL usando PyMySQL**

**import** pymysql

# Estabelecendo a conexão

conexao **=** pymysql**.***connect***(**

host**=**'localhost'**,** # ou IP do servidor

user**=**'root'**,** # usuário do banco

password**=**'sua\_senha'**,** # senha do banco

database**=**'loja\_virtual' # nome do banco de dados

**)**

**print(**"Conectado com sucesso!"**)**

# Encerrando a conexão

conexao**.***close***()**

Como conectar ao MySQL usando PyMySQL

**Executando comandos SQL com PyMySQL**

Para interagir com o banco, usamos um **cursor**, que é o objeto responsável por enviar comandos SQL:

**import** pymysql

# Conectando

conexao **=** pymysql**.***connect***(**

host**=**'localhost'**,**

user**=**'root'**,**

password**=**'sua\_senha'**,**

database**=**' loja\_virtual'

**)**

# Criando o cursor

cursor **=** conexao**.***cursor***()**

# Criando uma tabela (exemplo didático)

cursor**.***execute***(**"""

CREATE TABLE IF NOT EXISTS alunos (

id INT AUTO\_INCREMENT PRIMARY KEY,

nome VARCHAR(100),

nota FLOAT

)

"""**)**

# Inserindo dados

cursor**.***execute***(**"INSERT INTO alunos (nome, nota) VALUES (%s, %s)"**,** **(**'Ana'**,** 9.2**))**

# Confirmando a transação (importante!)

conexao**.***commit***()**

# Encerrando conexões

cursor**.***close***()**

conexao**.***close***()**

**Explicando os pontos importantes:**

| **Elemento** | **Função** |
| --- | --- |
| pymysql.connect() | Abre a conexão com o banco. |
| cursor() | Objeto que executa comandos SQL. |
| cursor.execute() | Executa uma instrução SQL. |
| commit() | Confirma as mudanças (INSERT, UPDATE, DELETE). |
| close() | Fecha a conexão ou cursor, liberando recursos. |
| %s | Placeholder seguro para inserir valores e evitar SQL Injection. |

## **Analogias para facilitar o aprendizado**

* **Conexão** → é como ligar um fio entre seu programa Python e o MySQL.
* **Cursor** → é como um "lápis" que escreve e lê os dados do banco.
* **Commit** → é como salvar um documento: as mudanças só ficam permanentes depois disso.
* **Placeholders (%s)** → são como “espaços em branco” seguros onde você encaixa os valores.

**Exemplo com leitura de dados (SELECT):**

cursor **=** conexao**.***cursor***()**

cursor**.***execute***(**"SELECT \* FROM alunos"**)**

resultados **=** cursor**.***fetchall***()**

**for** linha **in** resultados**:**

**print(**f"ID: {linha**[**0**]**}, Nome: {linha**[**1**]**}, Nota: {linha**[**2**]**}"**)**

**Segurança: Nunca concatene strings diretamente!**

❌ **Errado**:

cursor**.***execute***(**f"INSERT INTO alunos (nome, nota) VALUES ('{nome}', {nota})"**)**

✅ **Certo**:

cursor**.***execute***(**"INSERT INTO alunos (nome, nota) VALUES (%s, %s)"**,** **(**nome**,** nota**))**

**Usar placeholders evita ataques de SQL Injection, uma falha de segurança grave.**

**Para memorizar:**

* O PyMySQL permite conectar o Python ao MySQL de forma fácil e segura.
* Usamos a função connect para abrir conexão e cursor para executar comandos.
* É fundamental **fechar conexões** e **usar placeholders** para segurança.
* Com PyMySQL, podemos automatizar a leitura, gravação e atualização de dados no banco.