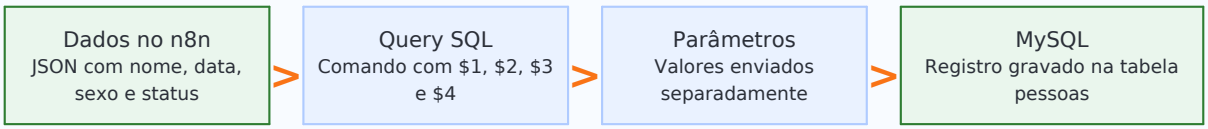
EXECUTE SQL NO N8N

GUIA PRÁTICO DE SQL INJECTION E PARÂMETROS NO N8N

Como proteger melhor suas consultas SQL usando boas práticas com Query Parameters



Objetivo do material

Entender por que parâmetros tornam suas consultas mais organizadas, mais seguras e mais profissionais ao trabalhar com MySQL dentro do n8n.

Segurança

Separe a estrutura SQL dos dados recebidos pelo fluxo.

Prática

Use $1, $2, $3 e $4 no Execute SQL Query.

Validação

Confira o retorno success true no n8n.

Conferência

Confirme o registro final no phpMyAdmin.

Material de apoio - SQL PRÓ para n8n

1. O que você vai aprender

Você já aprendeu a gravar dados na tabela pessoas usando o comando INSERT. Agora chegou o momento de dar um passo mais profissional: executar consultas SQL usando parâmetros.

Quando usamos SQL dentro de automações, precisamos tomar cuidado com a forma como os dados entram na consulta. Não basta fazer funcionar. Também precisamos fazer de uma forma mais segura e organizada.

Ideia central

O SQL diz o que deve ser feito. Os parâmetros dizem quais valores serão usados.

Você vai entender	Na prática
O que é SQL Injection	Por que misturar dados diretamente no SQL pode ser perigoso.
O que são parâmetros	Como usar $1, $2, $3 e $4 para receber valores.
Como configurar no n8n	Onde adicionar Query Parameters no node Execute SQL Query.
Como validar o resultado	Como conferir success true e o registro no phpMyAdmin.

2. Relembrando o INSERT na tabela pessoas

A tabela pessoas recebe dados como nome, data de nascimento, sexo e status. Em um exemplo simples, poderíamos escrever os valores diretamente dentro da consulta.

EXEMPLO DIDÁTICO COM VALORES DIRETOS

```
INSERT INTO pessoas (nome, data_nascimento, sexo, status)
VALUES ('Rafael Souza Santos', '1995-11-05', 'Masculino', 1);
```

Esse modelo ajuda no aprendizado, mas em automações reais os valores geralmente vêm de fora: formulários, webhooks, APIs, planilhas, outros nodes do n8n ou sistemas externos.

Ponto importante

Quando os valores vêm de fora da query, é melhor evitar misturar esses dados diretamente dentro do texto SQL.

Campo	Valor do exemplo	Observação
nome	Rafael Souza Santos	Texto que vem do fluxo do n8n.
data_nascimento	1995-11-05	Data no formato usado pelo MySQL.
sexo	Masculino	Valor compatível com o ENUM da tabela.
status	1	Valor numérico usado para pessoa ativa.

3. O que é SQL Injection

SQL Injection é uma técnica de ataque em que alguém tenta enviar comandos SQL maliciosos dentro de campos que deveriam receber apenas dados comuns.

É como se o sistema esperasse receber apenas um nome, mas alguém tentasse enviar um comando para manipular o banco.

O sistema espera receber	Mas alguém poderia tentar enviar
Rafael Souza Santos	Rafael'; DROP TABLE pessoas; --

Atenção

O exemplo acima é apenas didático. A ideia é mostrar o risco de permitir que valores externos se misturem livremente com a estrutura do SQL.

O problema acontece quando o banco não consegue diferenciar com clareza o que é comando SQL e o que é apenas valor enviado pelo usuário.

Comando SQL

É a instrução que manda o banco fazer algo, como INSERT, SELECT, UPDATE ou DELETE.

Valor recebido

É o dado que deve ser gravado ou consultado, como nome, email, data ou status.

4. Por que SQL Injection é perigoso

Uma consulta SQL montada sem cuidado pode permitir que dados externos interfiram no comportamento do banco. Em uma automação, isso pode gerar desde pequenos erros até problemas graves de segurança.

Consulta indevida

Alguém pode tentar acessar informações que não deveria ver.

Alteração de dados

Registros podem ser modificados de forma indevida.

Exclusão de dados

Dados importantes podem ser apagados ou comprometidos.

Falhas no fluxo

A automação pode quebrar ou gravar dados incorretos.

Exposição de dados

Informações sensíveis podem ficar vulneráveis.

Perda de confiança

O sistema deixa de parecer confiável e profissional.

Resumo simples

SQL Injection não é apenas um erro técnico. É uma falha de segurança que pode comprometer dados importantes.

5. Consulta menos segura: valores dentro do SQL

Em aulas iniciais, é comum ver consultas em que os valores do n8n são colocados diretamente dentro da query. Esse modelo pode funcionar, mas mistura a estrutura SQL com os dados vindos do fluxo.

FUNCIONA, MAS NÃO É A MELHOR PRÁTICA

```
INSERT INTO pessoas (nome, data_nascimento, sexo, status)
VALUES ('{{ $json.nome }}', '{{ $json.data_nascimento }}', '{{ $json.Sexo }}', {{ $json.status }});
```

Problema	Por que merece atenção
Dados dentro da query	Os valores ficam misturados com o comando SQL.
Mais chance de erro	Aspas, formato de data ou caracteres especiais podem quebrar a consulta.
Menos organizado	A query fica mais difícil de manter e revisar.
Menos profissional	Não segue a boa prática de separar comando e valores.

Frase-chave

Funcionar não significa necessariamente ser a melhor forma.

6. O que são parâmetros em SQL

Parâmetros são espaços reservados dentro da consulta SQL. Eles indicam que um valor será enviado separadamente.

CONSULTA COM PLACEHOLDERS

```
INSERT INTO pessoas (nome, data_nascimento, sexo, status)
VALUES ($1, $2, $3, $4);
```

Nesse modelo, a query mantém apenas a estrutura do comando. Os valores reais são enviados fora da consulta, na área Query Parameters do n8n.

Placeholder	O que significa
$1	Espaço reservado para o primeiro valor.
$2	Espaço reservado para o segundo valor.
$3	Espaço reservado para o terceiro valor.
$4	Espaço reservado para o quarto valor.

Definição simples

O placeholder não é o valor. Ele é apenas o lugar onde o valor será encaixado com mais segurança.

7. Entendendo a correspondência entre campos e parâmetros

A ordem é o ponto mais importante. O primeiro placeholder recebe o primeiro valor informado em Query Parameters, o segundo recebe o segundo valor, e assim por diante.

QUERY PRINCIPAL

```
INSERT INTO pessoas (nome, data_nascimento, sexo, status)
VALUES ($1, $2, $3, $4);
```

Campo da tabela	Placeholder	Valor vindo do n8n
nome	$1	{{ $json.nome }}
data_nascimento	$2	{{ $json.data_nascimento }}
sexo	$3	{{ $json.Sexo }}
status	$4	{{ $json.status }}

Regra de ouro

A ordem dos valores em Query Parameters precisa bater com a ordem dos placeholders usados na consulta.

$1

Recebe o primeiro valor: nome.

$2

Recebe o segundo valor: data_nascimento.

$3

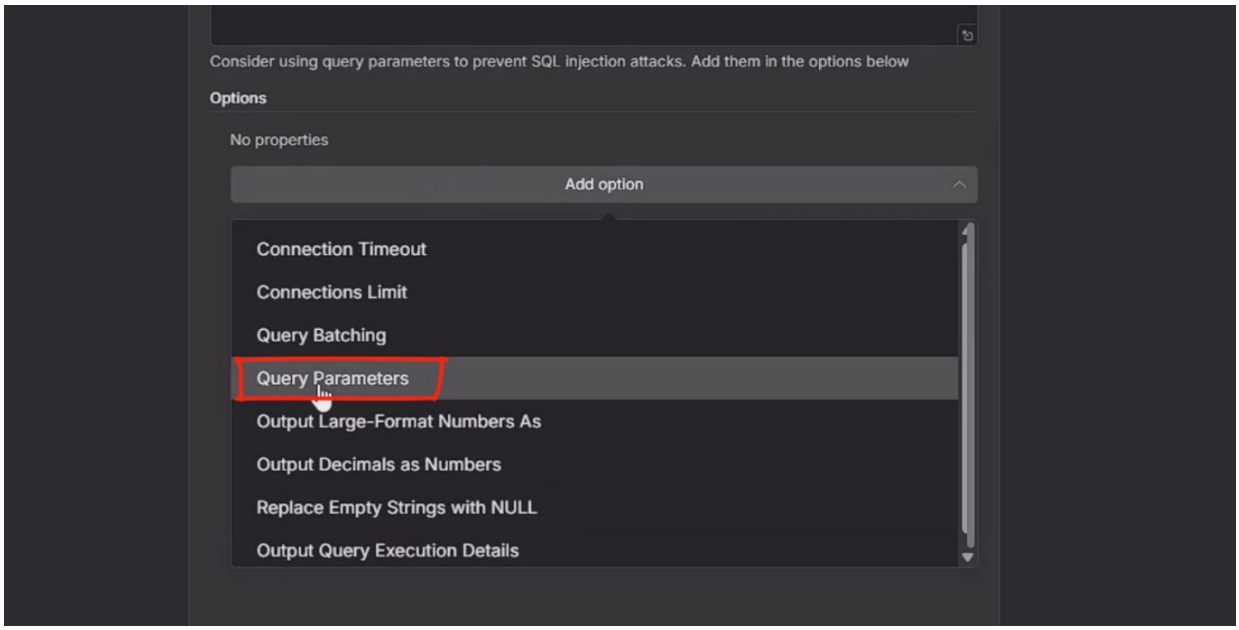
Recebe o terceiro valor: Sexo.

$4

Recebe o quarto valor: status.

8. Como adicionar Query Parameters no n8n

No node Execute SQL Query, os parâmetros ficam dentro da área Options. O aluno deve clicar em Add option e escolher Query Parameters.



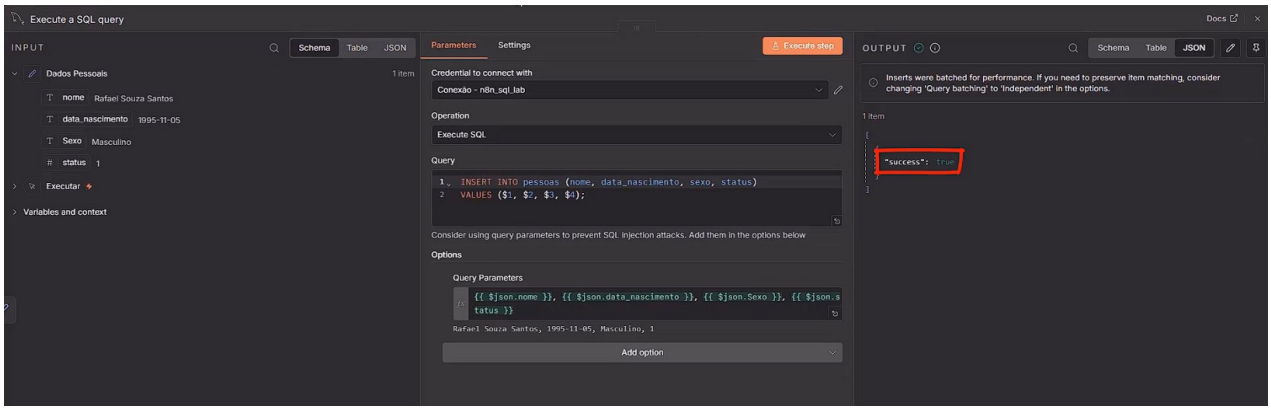
Passo	O que fazer
1	Abra o node Execute SQL Query.
2	Role até a seção Options.
3	Clique em Add option.
4	Selecione Query Parameters.
5	Informe os valores na mesma ordem dos placeholders.

Dica

Use Query Parameters sempre que os valores vierem do JSON, de formulários, webhooks, APIs ou qualquer fonte externa.

9. Executando o INSERT com parâmetros

Agora veja o exemplo completo: a query usa $1, $2, $3 e $4, enquanto os valores reais são informados em Query Parameters.



QUERY

```
INSERT INTO pessoas (nome, data_nascimento, sexo, status)
VALUES ($1, $2, $3, $4);
```

QUERY PARAMETERS

```
{{ $json.nome }},
{{ $json.data_nascimento }},
{{ $json.Sexo }},
{{ $json.status }}
```

Resultado esperado

Quando a execução funciona, o OUTPUT do n8n pode mostrar { "success": true }. Isso indica que o comando foi executado com sucesso.

10. Como interpretar o retorno success true

Após executar o node, o n8n retorna uma resposta indicando o resultado da operação. No exemplo da aula, o retorno foi:

OUTPUT DO N8N

```
{
  "success": true
}
```

Esse retorno indica que o comando INSERT foi aceito e executado pelo banco de dados.

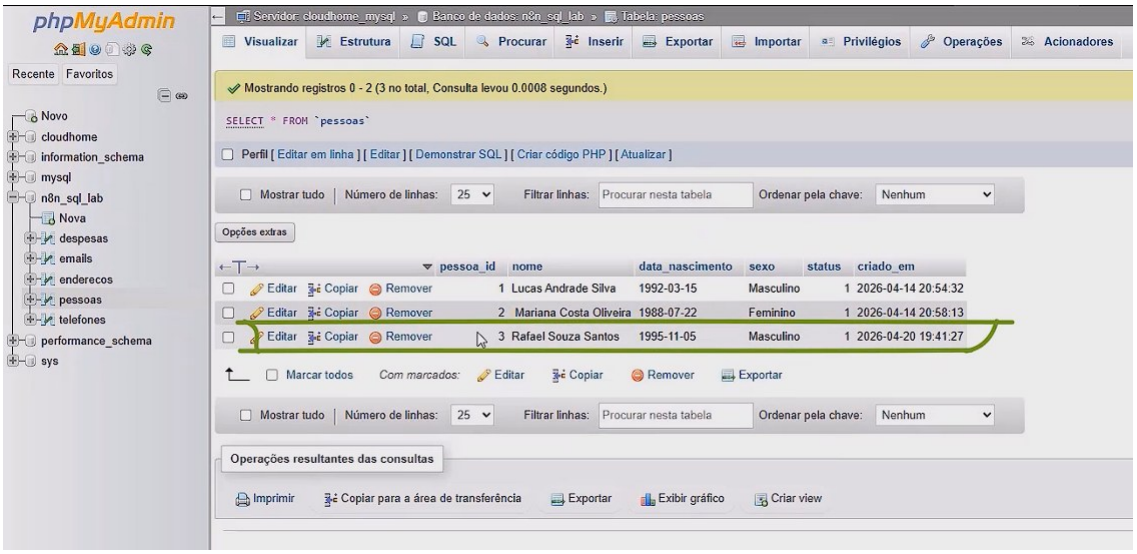
Importante

success: true confirma que a query foi executada. Mesmo assim, a boa prática é conferir no phpMyAdmin se o registro apareceu na tabela correta.

Retorno	Significado	Próximo passo
success: true	O INSERT foi executado.	Abrir a tabela pessoas no phpMyAdmin.
Erro de sintaxe	A query pode ter sido escrita incorretamente.	Revisar campos, vírgulas, parênteses e placeholders.
Erro de parâmetro	Faltou valor ou a ordem está incorreta.	Conferir Query Parameters.

11. Conferindo o registro no phpMyAdmin

Depois da execução no n8n, abra a tabela pessoas no phpMyAdmin para confirmar se o registro foi gravado.



Campo	Valor observado
pessoa_id	3 - gerado automaticamente pelo banco
nome	Rafael Souza Santos
data_nascimento	1995-11-05
sexo	Masculino
status	1
criado_em	Data e hora geradas pelo banco

Teste completo

A execução só fica totalmente validada quando o aluno confirma no phpMyAdmin que o novo registro foi realmente inserido.

12. Comparação: valores diretos vs parâmetros

A diferença principal está na separação entre a estrutura da consulta e os valores recebidos pelo fluxo.

Modelo	Como fica	Quando usar
Valores diretos	Os valores entram dentro do texto SQL.	Útil apenas para entender a lógica em exemplos simples.
Parâmetros	A query usa placeholders e os valores ficam em Query Parameters.	Melhor prática para automações reais.

MENOS RECOMENDADO EM AUTOMAÇÕES REAIS

```
INSERT INTO pessoas (nome, data_nascimento, sexo, status)
VALUES ('{{ $json.nome }}', '{{ $json.data_nascimento }}', '{{ $json.Sexo }}', {{ $json.status }});
```

MAIS PROFISSIONAL

```
INSERT INTO pessoas (nome, data_nascimento, sexo, status)
VALUES ($1, $2, $3, $4);
```

Melhor prática

Use parâmetros sempre que for trabalhar com valores dinâmicos vindos do n8n, formulários, webhooks, APIs ou qualquer fonte externa.

13. Erros comuns ao usar parâmetros

Parâmetros ajudam muito, mas precisam ser usados corretamente. Veja os erros mais comuns que iniciantes cometem.

1. Esquecer Query Parameters

A query usa $1, $2, $3 e $4, mas os valores não são informados na área de parâmetros.

2. Trocar a ordem

O primeiro parâmetro vai para $1. Se a ordem for trocada, os dados podem ir para campos errados.

3. Usar placeholder inexistente

A query pede $5, mas só existem quatro valores informados.

4. Informar valores demais

A query usa quatro placeholders, mas o aluno informa cinco valores.

EXEMPLO DE ERRO

```
-- Ordem errada nos parâmetros:
{{ $json.data_nascimento }}, {{ $json.nome }}, {{ $json.Sexo }}, {{ $json.status }}

-- Resultado provável:
-- a data tentaria entrar no campo nome
-- o nome tentaria entrar no campo data_nascimento
```

Regra prática

Quando algo der errado, confira primeiro se a ordem dos placeholders bate com a ordem dos valores em Query Parameters.

14. Mais cuidados importantes

Além da ordem dos parâmetros, existem outros detalhes que merecem atenção quando trabalhamos com SQL dentro do n8n.

Erro	Exemplo	Como evitar
Nome do campo JSON errado	{{ $json.sexo }} quando o campo real é {{ $json.Sexo }}	Confira maiúsculas e minúsculas no JSON.
Achar que parâmetro substitui validação	Enviar data em formato inválido	Valide formato de data, status e valores permitidos.
Usar parâmetros para nomes de tabela	$1 no lugar de pessoas	Use parâmetros para valores, não para nomes de tabelas ou colunas.
Não conferir no phpMyAdmin	Ver apenas success true	Confirme o registro gravado na tabela.

Muito importante

Parâmetros ajudam na segurança, mas não substituem validação. O ideal é usar parâmetros e também conferir se os dados recebidos fazem sentido.

REGRA DE USO

```
-- Parâmetros são para valores:
VALUES ($1, $2, $3, $4)

-- Não são para nomes de tabela ou coluna:
-- SELECT * FROM $1;  -- evite esse tipo de uso
```

15. Boas práticas para proteger consultas SQL no n8n

Ao trabalhar com SQL no n8n, siga um padrão simples: query organizada, parâmetros na ordem correta, dados validados e conferência final no banco.

Use parâmetros

Sempre que os valores vierem de fora da query.

Separe SQL dos dados

A query define a ação. Os parâmetros enviam os valores.

Confira a ordem

$1 recebe o primeiro valor, $2 recebe o segundo e assim por diante.

Valide o JSON

Confirme se nome, data, sexo e status existem antes de executar.

Teste com poucos registros

Valide o fluxo antes de automatizar em escala.

Confirme no phpMyAdmin

Verifique se o registro entrou na tabela correta.

Resumo profissional

Parâmetros não deixam sua automação perfeita, mas já colocam sua consulta em um nível muito mais seguro e profissional.

16. Checklist antes de executar SQL com parâmetros

Use este checklist antes de executar uma consulta SQL com dados dinâmicos no n8n.

☐ Estou usando a tabela correta?	☐ O campo sexo usa valor permitido pelo ENUM?
☐ Conferi os campos da tabela pessoas?	☐ O campo status recebe um número válido?
☐ Usei placeholders na query?	☐ Conferi se o JSON contém os dados esperados?
☐ Usei $1, $2, $3 e $4 na ordem correta?	☐ Executei o node com atenção?
☐ Adicionei a opção Query Parameters?	☐ Verifiquei o retorno success true?
☐ A ordem dos parâmetros bate com a ordem dos campos?	☐ Conferi o registro no phpMyAdmin?
☐ O campo nome recebe o valor correto?	☐ Entendi que parâmetros são usados para valores?
☐ A data está no formato YYYY-MM-DD?	☐ Validei dados externos antes de gravar?

Use como rotina

Esse checklist ajuda a evitar erros simples, principalmente quando a automação começa a receber dados reais de várias fontes.

17. Resumo rápido - cola de consulta

Esta página serve como consulta rápida para o aluno revisar antes de executar um INSERT com parâmetros.

Conceito	Para que serve	Exemplo
SQL Injection	Tentativa de inserir SQL malicioso em campos de entrada.	Rafael'; DROP TABLE pessoas; --
Placeholder	Espaço reservado para um valor.	$1
Query Parameters	Local onde os valores são informados no n8n.	{{ $json.nome }}
INSERT INTO	Comando para inserir dados.	INSERT INTO pessoas (...)
VALUES	Define os valores que serão gravados.	VALUES ($1, $2, $3, $4)
success true	Indica execução bem-sucedida.	{ "success": true }

MODELO FINAL

```
-- Query:
INSERT INTO pessoas (nome, data_nascimento, sexo, status)
VALUES ($1, $2, $3, $4);

-- Query Parameters:
{{ $json.nome }},
{{ $json.data_nascimento }},
{{ $json.Sexo }},
{{ $json.status }}
```

Frase para lembrar

Use parâmetros sempre que os valores vierem de fora da query.

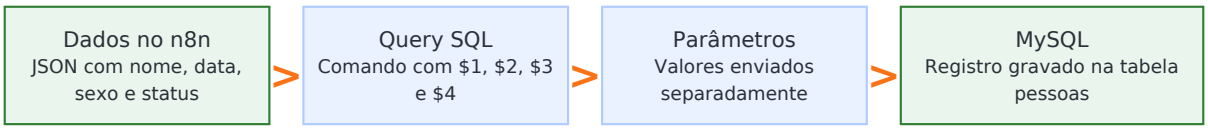
18. Encerramento

Usar parâmetros é um passo importante para criar automações mais seguras e profissionais.

A partir de agora, o aluno não está apenas gravando dados no MySQL. Ele está aprendendo a gravar dados com mais cuidado, organização e segurança.

Mensagem principal

Esse é um dos pontos que diferencia uma automação improvisada de uma automação mais profissional.



Antes	Agora
Valores misturados dentro da query	Estrutura SQL separada dos valores
Mais risco de erro	Consulta mais organizada
Menos clareza na manutenção	Parâmetros mais fáceis de revisar
Apenas funciona	Funciona com mais segurança e profissionalismo

Material de apoio - EXECUTE SQL NO N8N

SQL Injection e boas práticas com parâmetros