# Ascensão do Mestre SQL

## Do Padawan de Dados ao Mestre das Consultas MySQL


JOBER JUNIOR

# Introdução

Este ebook foi preparado para guiá-lo, sem rodeios, do nível iniciante ao domínio das consultas SQL em MySQL. Cada página apresenta um comando, explica seu propósito em um parágrafo objetivo e demonstra com exemplos pragmáticos inspirados no universo de Star Wars. Ao final, você terá a base necessária para extrair dados de qualquer banco de forma eficiente.


# 01

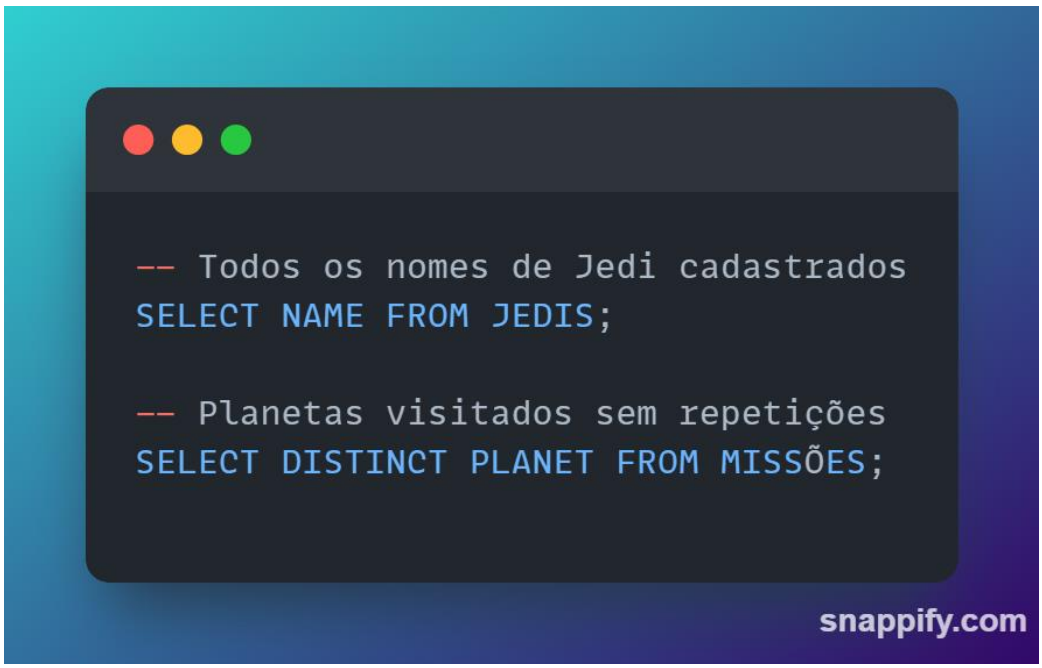
# Consultas Básicas

# SELECT E SELECT DISTINCT

O comando `SELECT` é a porta de entrada para ler dados em um banco: você escolhe colunas específicas ou usa * para todas elas.

Já o `SELECT DISTINCT` elimina linhas duplicadas, retornando apenas valores únicos em uma coluna.

Imagine uma tabela `JEDIS` com várias missões repetidas: para listar todos os nomes de planetas já visitados sem repetições, use `SELECT DISTINCT PLANET`



# WHERE, AND, OR

Para filtrar registros, `WHERE` é indispensável: ele aplica uma condição para selecionar apenas as linhas que interessam.

Quando múltiplos critérios se aplicam, `AND` exige que ambos sejam verdadeiros; `OR` aceita qualquer um.

Por exemplo, em uma tabela `DROIDS`, podemos buscar todos os astromecânicos ativos em ‘Tatooine’ e de modelo ‘R2’:

# IN e NOT IN

Quando temos uma lista de valores possível, `IN` simplifica várias comparações em uma só expressão.

`NOT IN` faz o inverso: exclui aqueles valores.

Em `PILOTOS`, para encontrar quem já voou nas três principais rotas você escreveria `WHERE ROUTE IN ('Corellia', 'Naboo', 'Alderaan')`.

```sql
-- Pilotos que voaram em rotas principais
SELECT *
FROM PILOTOS
WHERE ROUTE IN ('Corellia', 'Naboo', 'Alderaan');

-- Pilotos que NÃO voaram em rotas principais
SELECT *
FROM PILOTOS
WHERE ROUTE NOT IN ('Corellia', 'Naboo', 'Alderaan');
```

snappify.com

# IN e NOT IN

Quando temos uma lista de valores possível, `IN` simplifica várias comparações em uma só expressão.

`NOT IN` faz o inverso: exclui aqueles valores.

Em `PILOTOS`, para encontrar quem já voou nas três principais rotas você escreveria `WHERE ROUTE IN ('Corellia', 'Naboo', 'Alderaan')`.

```sql
-- Pilotos que voaram em rotas principais
SELECT *
FROM PILOTOS
WHERE ROUTE IN ('Corellia', 'Naboo', 'Alderaan');

-- Pilotos que NÃO voaram em rotas principais
SELECT *
FROM PILOTOS
WHERE ROUTE NOT IN ('Corellia', 'Naboo', 'Alderaan');
```

snappify.com

# ORDER BY

Por padrão, o banco devolve linhas em ordem arbitrária.

`ORDER BY` impõe uma sequência, seja ascendente (`ASC`) ou descendente (`DESC`).

Em `SITHS`, para listar do mais antigo ao mais recente, ordene pela data de queda:

# 02

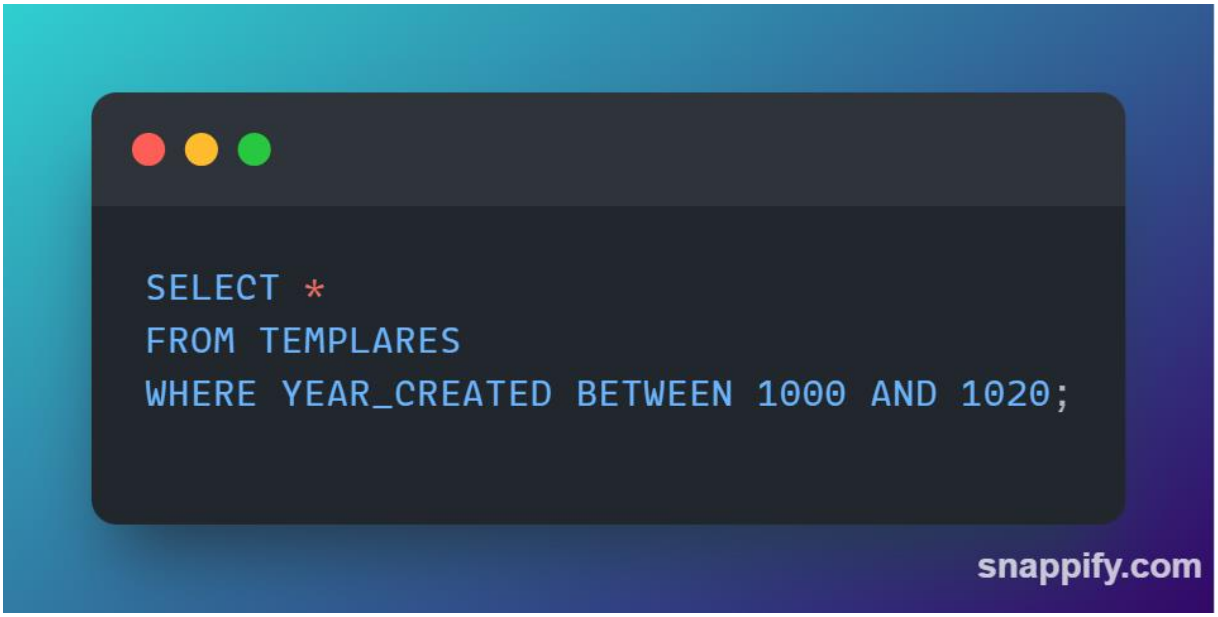
# Consultas Avançadas

# BETWEEN

O comando BETWEEN serve para filtrar os dados dentro de um determinado escopo.

Quando precisamos de um intervalo contínuo, `BETWEEN` é direto e legível.

Em `TEMPLARES`, para selecionar todos os artefatos criados entre 1000 e 1020, usamos:



# LIKE

O comando LIKE é utilizado para buscar padrões de texto em consultas SQL

Para buscas textuais flexíveis, `LIKE` usa curingas:
% representa zero ou mais caracteres;
_ representa exatamente um.

Em `CÓDIGOS`, para encontrar todas as chaves que começam com ‘AB-’:

```sql
--Codigos que começam com AB e é seguida
 por qualquer sequência de caracteres.
SELECT * FROM CÓDIGOS
WHERE KEYCODE LIKE 'AB-%';

-- Codigos que começam com 'Z7' e terminem com 'a'
SELECT * FROM CÓDIGOS
WHERE KEYCODE LIKE 'Z7%a';
```

snappify.com

# LIMIT E OFFSET

Quando a consulta retorna muitas linhas, `LIMIT` restringe o resultado aos primeiros N registros;

`OFFSET` pula os primeiros M. Útil para paginação em interfaces de usuário.

Em `RELATÓRIOS`, para mostrar as linhas 11 a 20:

```sql
-- Mostra 10 Relatórios começando do 11º.
SELECT *
FROM RELATÓRIOS
LIMIT 10 OFFSET 10;
```

snappify.com

# IS NULL e IS NOT NULL

Dados ausentes ou não registrados aparecem como `NULL`.

Para encontrá-los, use `IS NULL`;
para garantir que há valor, `IS NOT NULL`.

Em `MISSÕES`, descobrimos quais ainda não têm data de retorno definida:

```sql
-- Mostra missões sem data de retorno.
SELECT MISSION_NAME
FROM MISSÕES
WHERE RETURN_DATE IS NULL;

-- Mostra Missões com data de retorno.
SELECT MISSION_NAME
FROM MISSÕES
WHERE RETURN_DATE IS NOT NULL;
```

snappify.com

# Aliases (AS)

Para tornar consultas complexas mais legíveis, você pode renomear colunas ou tabelas em tempo de execução com AS.

Por exemplo, em CRUZADORES e CAPITAIS, apresentamos o nome do cruzador como 'Nave_Mestra', ao mesmo tempo que nos referimos as tabelas 'CRUZADORES' e 'CAPITAIS' como 'c' e 'k' :

```sql
SELECT c.NAME AS 'Nave_Mestra', k.NAME AS 'Capital'
FROM CRUZADORES AS c
JOIN CAPITAIS AS k
  ON c.CAPITAL_ID = k.ID;
```

snappify.com

# 03

# Relacionamento de Tabelas

# INNER JOIN

Joins em bancos de dados são utilizados para combinar linhas de diferentes tabelas com base em uma coluna relacionada, permitindo obter informações completas.

O INNER JOIN combina registros de duas tabelas com base em uma condição de igualdade, selecionando somente os registros que possuem valores correspondentes em ambas as tabelas.

Em `PADAWANS` e `MESTRES`, podemos listar cada padawan com seu mestre atribuído:

```sql
SELECT p.NAME AS 'Padawan', m.NAME AS 'Mestre'
FROM PADAWANS AS p
INNER JOIN MESTRES AS m
  ON p.MESTRE_ID = m.ID;
```

snappify.com

# LEFT JOIN e RIGHT JOIN

LEFT JOIN retorna todos os registros da tabela à esquerda, mesmo que não haja correspondência na tabela à direita.

O RIGHT JOIN retorna todos osregistros da tabela à direita e os registros correspondentes da tabela à esquerda, quando existem, usando o SQL.

A ordem das tabelas no JOIN é crucial. Colocar a tabela à esquerda no LEFT JOIN primeiro garante que todos os seus registros sejam exibidos.

Para ver todos os mestres, indicando se têm padawans:

```sql
-- Todos os mestres e seus padawans (se houver)
SELECT m.NAME AS 'Mestre', p.NAME AS 'Padawan'
FROM MESTRES AS m
LEFT JOIN PADAWANS AS p ON p.MESTRE_ID = m.ID;

-- Todos os mestres e seus padawans (se houver)
SELECT p.NAME AS 'Padawan', m.NAME AS 'Mestre'
FROM PADAWANS AS p
RIGHT JOIN MESTRES AS m ON p.MESTRE_ID = m.ID;
```

snappify.com

# Conclusão

Você percorreu cada comando essencial, com exemplos práticos que simulam cenários variados. Agora, sem fantasias, você tem o conhecimento necessário para escrever consultas SQL poderosas em MySQL. A partir deste ponto, pratique, crie suas próprias queries, sem dúvidas dos resultados. Parabéns: você é agora um mestre em banco de dados.

Obrigado por ler este ebook.