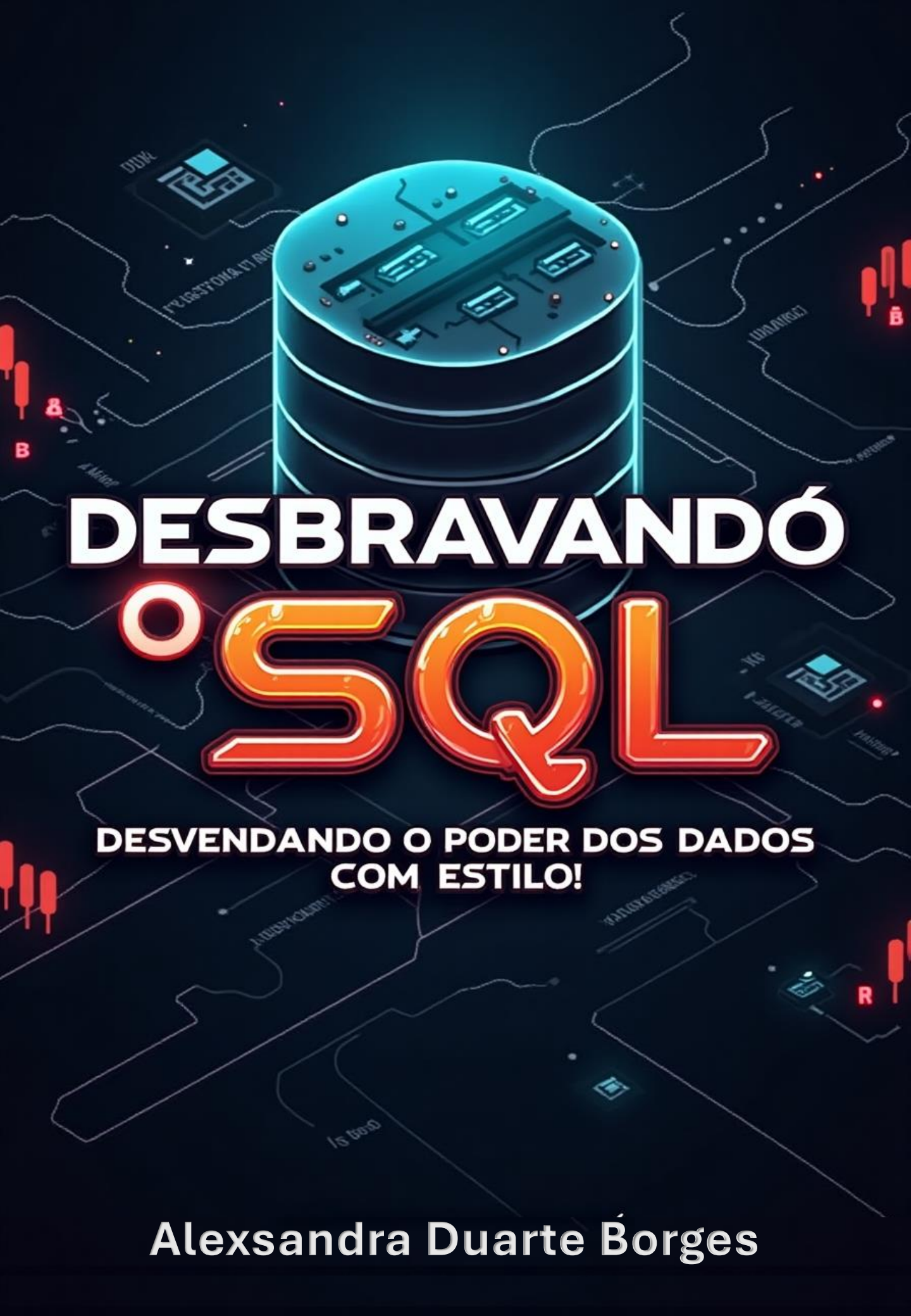
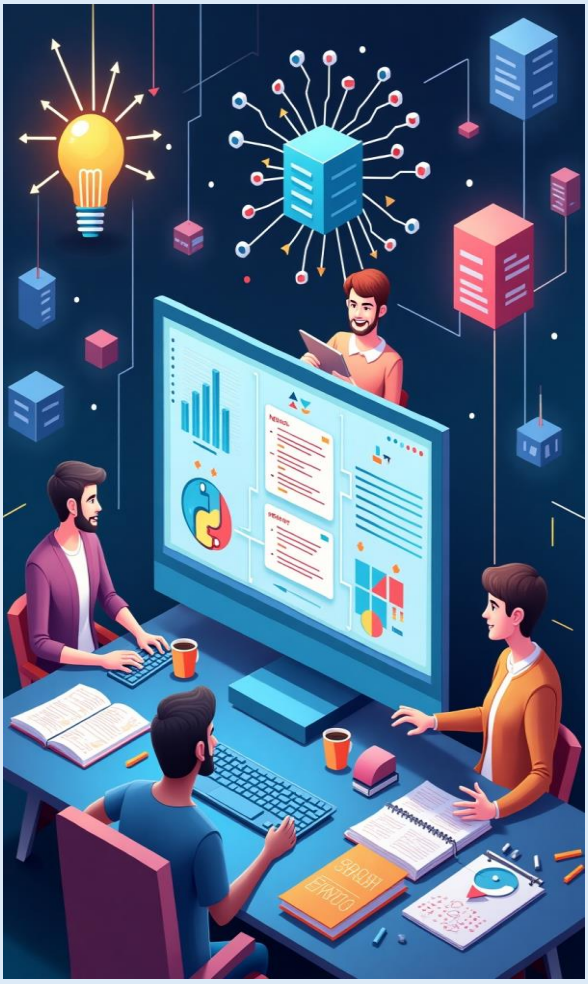

DESBRAVANDÓ
SQL
DESVENDANDO O PODER DOS DADOS
COM ESTILO!
Alexsandra Duarte Borges

Capítulo 1:
Principais Comandos de SQL



A linguagem SQL (*Structured Query Language*) é a chave para explorar, manipular e organizar dados em bancos de dados. Sem ela, seria como tentar encontrar uma agulha no palheiro sem o mapa. Imagine que você é um explorador em um vasto mundo de dados, e o SQL é sua espada afiada, capaz de cortar através da complexidade e revelar as informações que você precisa. Vamos começar com os principais comandos dessa linguagem poderosa.

Um guia
para **os principais comandos SQL**

1.1 SELECT: O Portal para os Dados

O comando SELECT é o mais fundamental e essencial quando se fala em SQL. Ele é como uma lente de aumento que nos permite visualizar os dados contidos nas tabelas do banco. Usando SELECT, você pode escolher exatamente quais colunas de dados deseja consultar.

Exemplo simples:

```
SELECT nome, idade, cidade FROM clientes;
```

Neste caso, estamos pedindo ao SQL para nos mostrar os campos nome, idade e cidade da tabela clientes. É como fazer uma visita à biblioteca e pegar apenas os livros que contêm as informações que você quer – nada mais, nada menos. Mas nesta caso, como não limitamos o número de registros do resultado, ele irá trazer todos os registros da tabela clientes.

1.2 WHERE: Filtrando Informações

Agora que conseguimos olhar para os dados com o SELECT, muitas vezes precisamos filtrar ou limitar os resultados. O comando WHERE é nossa ferramenta de precisão, nos ajudando a trazer apenas os dados que atendem a um determinado critério.

Exemplo prático:

```
SELECT nome, idade, cidade FROM clientes WHERE idade > 18;
```

Aqui, estamos dizendo ao SQL: "Mostre-me apenas os clientes cuja idade seja superior a 18 anos." É como pedir para o assistente pessoal trazer apenas os itens relevantes em uma loja – nada de perder tempo com o que não importa!

1.3 INSERT INTO: Adicionando Dados

O comando INSERT INTO é como um superpoder que permite adicionar novas informações a uma tabela. Se os dados são um tesouro, o INSERT INTO é o seu baú onde você armazena as descobertas.

Exemplo de como adicionar dados a uma tabela:

```
INSERT INTO clientes (nome, idade, cidade)
VALUES ('Maria', 30, 'São Paulo');
```

Aqui, estamos dizendo ao SQL para adicionar um novo cliente, chamado Maria, de 30 anos, da cidade de São Paulo, à tabela clientes. Simples, eficaz e poderoso!

1.4 UPDATE: Atualizando Dados

Com o comando UPDATE, você pode modificar dados existentes. Esse comando é útil quando você percebe que algo no banco de dados precisa ser corrigido ou alterado.

Exemplo de atualização de dados:

```
UPDATE clientes
SET idade = 31
WHERE nome = 'Maria';
```

Agora, estamos dizendo ao SQL: "Maria tem agora 31 anos, não 30." Isso pode ser crucial, pois garantir a precisão dos dados é o coração de qualquer aplicação!

1.5 DELETE: Removendo Dados

Às vezes, você precisa eliminar registros de uma tabela, seja por questões de limpeza de dados ou porque a informação não é mais relevante. O comando DELETE é a ferramenta para isso.

Exemplo de remoção de dados:

```
DELETE FROM clientes WHERE nome = 'Maria';
```

Aqui, estamos apagando o registro de Maria da tabela. Isso pode ser feito com segurança, desde que você tenha certeza de que os dados não são mais necessários. Esse comando funciona como uma borracha, apagando registros desnecessários ou obsoletos.

1.6 ORDER BY: Organizando os Resultados

Os dados podem estar armazenados em qualquer ordem, mas frequentemente precisamos de uma organização lógica. O comando ORDER BY ajuda a ordenar os dados de acordo com uma ou mais colunas, seja de forma crescente ou decrescente.

Exemplo de organização:

```
SELECT nome, idade FROM clientes ORDER BY idade DESC;
```

Neste exemplo, estamos dizendo ao SQL para exibir os nomes e idades dos clientes, mas organizados por idade de forma decrescente. Como organizar as peças de um quebra-cabeça para formar uma imagem mais clara!

1.7 GROUP BY: Agrupando Dados

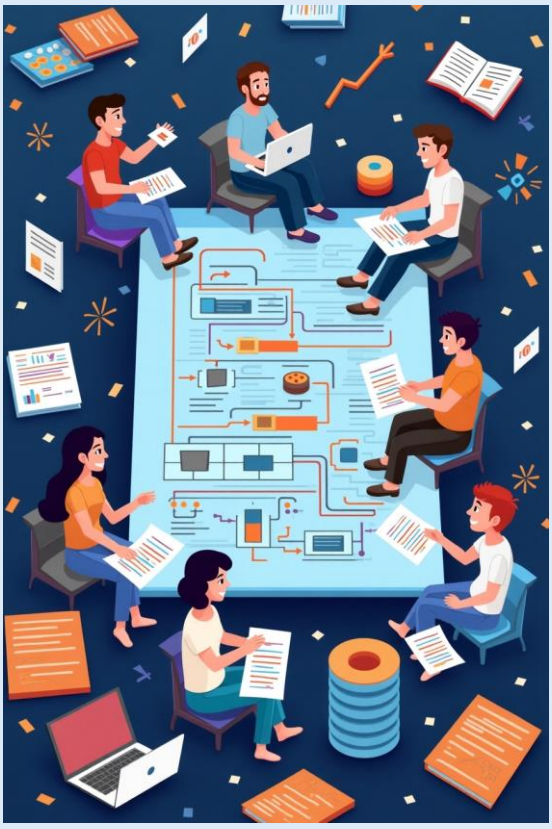
Imagine que você tem um grande número de dados e quer agrupá-los de acordo com um critério específico, como somar todas as vendas por mês ou contar quantos clientes há em cada cidade. O comando GROUP BY é ideal para isso.

Exemplo de agrupamento de dados:

```
SELECT cidade, COUNT(*) FROM clientes GROUP BY cidade;
```

Aqui, estamos pedindo ao SQL para contar quantos clientes existem em cada cidade. Esse comando é como um organizador de dados, que agrupa tudo em categorias fáceis de entender.

Capítulo 2:
Como Combinar os Comandos e Unir Dados de Mais de Uma Tabela



Quando estamos lidando com um banco de dados mais complexo, é muito comum que a informação esteja distribuída entre várias tabelas. Nesse cenário, precisamos aprender a combinar essas tabelas de maneira eficiente para obter uma visão completa. A junção de dados de múltiplas tabelas é onde a verdadeira mágica do SQL acontece. Vamos aprender como fazer isso com os comandos de JOIN!

Um guia
para **os principais comandos SQL**

2.1 JOIN: Unindo Tabelas

A palavra-chave JOIN é a ponte que conecta duas ou mais tabelas, criando um único conjunto de resultados. Existem vários tipos de junção, mas vamos explorar os mais comuns: INNER JOIN, LEFT JOIN e RIGHT JOIN.

2.1.1 INNER JOIN: A União Perfeita

O INNER JOIN é usado quando você quer combinar apenas os registros que possuem correspondência em ambas as tabelas. Em outras palavras, ele só trará os dados que têm uma conexão entre as tabelas.

Exemplo de uso:

```
SELECT clientes.nome, pedidos.valor
FROM clientes
INNER JOIN pedidos ON clientes.id =
pedidos.cliente_id;
```

Aqui, estamos unindo duas tabelas – clientes e pedidos – através do campo id de clientes e cliente_id de pedidos. O SQL vai retornar apenas os clientes que têm pedidos registrados. É como um evento de networking, onde só os convidados com convites em mãos entram na festa!

2.1.2 LEFT JOIN: Não Deixe Ninguém de Fora

O LEFT JOIN traz todos os registros da tabela à esquerda da junção, mesmo que não haja correspondência na tabela à direita. Se não houver correspondência, o SQL preencherá as colunas da tabela à direita com valores nulos.

Exemplo:

```
SELECT clientes.nome, pedidos.valor
FROM clientes
LEFT JOIN pedidos ON clientes.id = pedidos.cliente_id;
```

Neste caso, estamos trazendo todos os clientes, mesmo aqueles que não têm pedidos. Essa é uma maneira de garantir que ninguém seja deixado de fora, mesmo que não tenha feito compras recentemente!

2.1.3 RIGHT JOIN: A Visão Invertida

Da mesma forma que o LEFT JOIN, o RIGHT JOIN traz todos os registros da tabela à direita da junção, incluindo aqueles que não têm correspondência na tabela à esquerda.

Exemplo:

```
SELECT clientes.nome, pedidos.valor
FROM clientes
RIGHT JOIN pedidos ON clientes.id =
pedidos.cliente_id;
```

Aqui, estamos garantindo que todos os pedidos sejam exibidos, mesmo aqueles feitos por clientes que não estão registrados em nossa tabela de clientes. Imagine isso como uma abordagem inclusiva, onde todos os registros são considerados.

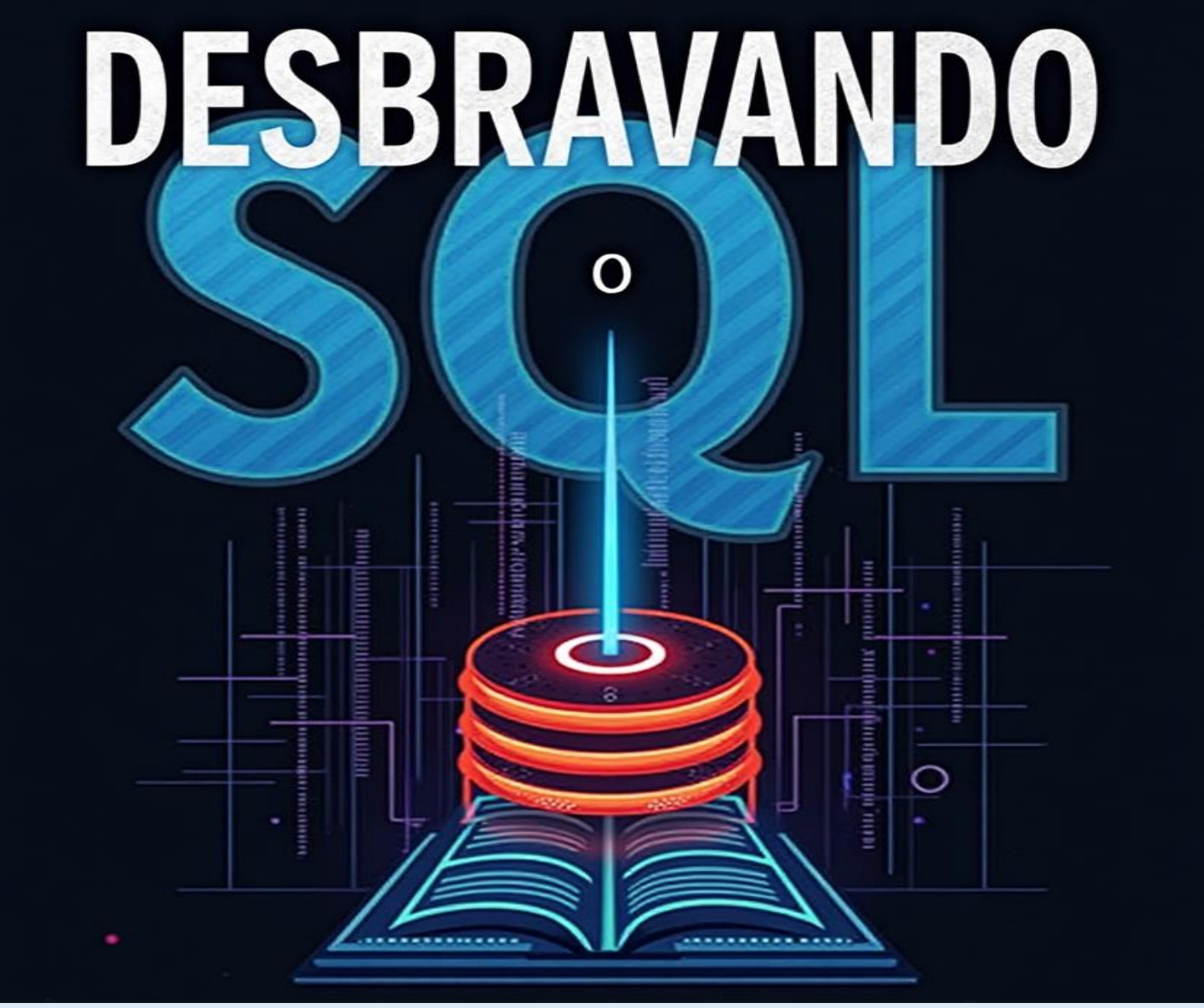
2.3 JOIN COM WHERE: Refinando a Junção

Assim como usamos o comando WHERE para filtrar dados, você pode combiná-lo com um JOIN para tornar suas consultas ainda mais específicas.

Exemplo:

```
SELECT clientes.nome, pedidos.valor
FROM clientes
INNER JOIN pedidos ON clientes.id =
pedidos.cliente_id
WHERE pedidos.valor > 100;
```

Aqui, estamos não apenas unindo as tabelas clientes e pedidos, mas também filtrando os pedidos cujo valor seja superior a 100. Isso torna a consulta mais eficaz e direcionada às suas necessidades.



Com esses conhecimentos, você já possui uma base sólida para começar a trabalhar com SQL e explorar o vasto universo dos dados. Lembre-se de que o SQL é uma ferramenta incrivelmente poderosa, capaz de transformar um oceano de informações em algo compreensível, acessível e útil. Agora, saia e desbrave o mundo dos dados com estilo!

Esse ebook foi gerado por IA e diagramado por humano.