Introdução ao SQL - SELECT e Subconsultas

Duração Total do Curso: 5 aulas (uma aula por semana)

Objetivo Geral: Ensinar aos alunos os conceitos fundamentais de SQL, com ênfase em SELECT e subconsultas, para que possam extrair dados de bancos de dados de forma eficiente e criar consultas complexas.

Aula 1: Introdução ao SQL e SELECT (2 horas)

- Introdução ao SQL e sua importância.
- Visão geral de bancos de dados relacionais.
- Comandos SQL básicos.
- Comando SELECT para consulta de dados.
- Cláusulas SELECT: FROM, WHERE, ORDER BY.
- Prática: Escrever consultas SELECT simples em um banco de dados de exemplo.

Aula 2: Filtragem de Dados e Operadores Lógicos (2 horas)

- Revisão da cláusula WHERE.
- Uso de operadores lógicos (AND, OR, NOT).
- Operadores de comparação.
- Consultas com várias condições.
- Prática: Escrever consultas SELECT com filtragem avançada.

Aula 3: Ordenação, Funções de Agregação e Grupos (2 horas)

- Revisão da cláusula ORDER BY.
- Funções de agregação (SUM, COUNT, AVG, MAX, MIN).
- Agrupamento de resultados com GROUP BY.

- Filtragem de grupos com HAVING.
- Prática: Realizar consultas com funções de agregação e agrupamento.

Aula 4: Subconsultas Simples (2 horas)

- Introdução às subconsultas.
- Subconsultas como expressões.
- Subconsultas em cláusulas WHERE.
- Subconsultas em cláusulas FROM.
- Prática: Criar consultas com subconsultas simples.

Aula 5: Subconsultas Correlacionadas e Subconsultas Aninhadas (2 horas)

- Subconsultas correlacionadas.
- Subconsultas aninhadas.
- Uso de subconsultas para resolver problemas complexos.
- Prática: Escrever consultas avançadas com subconsultas correlacionadas e aninhadas.

Avaliação Final (1 hora)

- Revisão dos conceitos aprendidos.
- Resolução de exercícios práticos.
- Avaliação escrita ou teste prático.
- Discussão de dúvidas e perguntas.

Recursos Necessários:

- Acesso a um sistema de gerenciamento de banco de dados (por exemplo, MySQL, PostgreSQL ou SQLite).
- Exemplos de bancos de dados de treinamento ou amostras.
- Material didático, incluindo apresentações de slides e documentação.
- Computadores para os alunos com software de banco de dados instalado.

Avaliação:

Os alunos serão avaliados com base na participação nas aulas, na conclusão de exercícios práticos e em uma avaliação final.

Este plano de curso proporcionará aos alunos uma compreensão sólida dos fundamentos do SQL, com ênfase em SELECT e subconsultas, preparando-os para criar consultas SQL eficazes e resolver problemas de análise de dados em bancos de dados relacionais.

sugestões de ambientes ou simuladores online que você pode usar para realizar práticas em cada aula do curso de SQL:

Aula 1: Introdução ao SQL e SELECT:

- SQLFiddle: Permite escrever consultas SQL online em diversos bancos de dados, como MySQL, PostgreSQL, e Oracle.
- W3Schools SQL Playground: Oferece um ambiente de prática interativo para consultas SQL.

Aula 2: Filtragem de Dados e Operadores Lógicos:

- SQLZoo: Fornece uma série de tutoriais interativos para aprender SQL, incluindo consultas com operadores lógicos.

Aula 3: Ordenação, Funções de Agregação e Grupos:

- SQLZoo: Continua sendo uma ótima opção para praticar funções de agregação e agrupamento.

Aula 4: Subconsultas Simples:

- SQLFiddle: Útil para criar consultas com subconsultas e verificar os resultados.

Aula 5: Subconsultas Correlacionadas e Subconsultas Aninhadas:

- SQLFiddle: Pode ser usado para práticas avançadas com subconsultas.
- HackerRank SQL: Oferece desafios SQL que incluem subconsultas e consultas complexas.

Lembre-se de que a escolha do ambiente ou simulador depende do sistema de gerenciamento de banco de dados que você está aprendendo (por exemplo, MySQL, PostgreSQL, SQL Server, etc.). Certifique-se de selecionar um ambiente que seja compatível com o SGBD que está sendo usado nas aulas.

Aula 1: Introdução ao SQL e SELECT

SQL e sua Importância

Nesta primeira aula, vamos mergulhar no mundo do SQL, que é a linguagem padrão para gerenciar e consultar bancos de dados relacionais. O SQL, que significa Structured Query Language (Linguagem de Consulta Estruturada), é uma habilidade essencial para qualquer pessoa envolvida no manuseio de dados. Desde cientistas de dados até desenvolvedores de software e administradores de banco de dados, o SQL é uma ferramenta fundamental para acessar, manipular e extrair informações de bancos de dados de maneira eficiente.

Visão Geral de Bancos de Dados Relacionais

Antes de começarmos a explorar o SQL em detalhes, é importante entender o contexto em que ele é usado - os bancos de dados relacionais. Bancos de dados relacionais são sistemas de armazenamento de dados que organizam informações em tabelas com linhas e colunas. Essas tabelas estão interconectadas por meio de chaves primárias e estrangeiras, permitindo a representação e a recuperação de dados de maneira estruturada.

Os bancos de dados relacionais são amplamente utilizados em empresas e organizações de todos os tamanhos devido à sua capacidade de garantir integridade dos dados, flexibilidade e eficiência.

O que é SQL?

O SQL é uma linguagem de programação utilizada para gerenciar bancos de dados relacionais. Bancos de dados relacionais são sistemas de armazenamento de dados que organizam informações em tabelas compostas por linhas e colunas. Essas tabelas são interconectadas por meio de chaves primárias e estrangeiras, permitindo uma representação estruturada dos dados e a recuperação de informações de forma eficaz.

Por que o SQL é Importante?

- Universalidade: O SQL é amplamente aceito e usado em uma variedade de sistemas de gerenciamento de banco de dados (SGBDs), como MySQL, PostgreSQL, SQL Server, Oracle, entre outros.

- Manipulação de Dados: Ele permite inserir, atualizar, excluir e consultar dados em bancos de dados. Essas operações são cruciais para qualquer aplicação que lide com informações.
- Recuperação de Dados: Através do comando SELECT, o SQL permite extrair dados específicos de uma ou mais tabelas, o que é fundamental para análise e relatórios.

Comandos SQL Básicos

Para começar nossa jornada no SQL, vamos explorar alguns comandos básicos:

SELECT: Usado para consultar dados de uma tabela.
FROM: Indica de qual tabela os dados devem ser selecionados.
WHERE: Permite filtrar os resultados com base em condições específicas.

Esses comandos formam a espinha dorsal das consultas SQL e são a base para recuperar informações de um banco de dados.

Comando SELECT para Consulta de Dados

O comando SELECT é a peça central das consultas SQL. Ele nos permite recuperar dados de uma ou mais tabelas. A estrutura básica de um comando SELECT é a seguinte:

SELECT coluna1, coluna2, ...

FROM nome_da_tabela

WHERE condição;

coluna1, coluna2, ...: As colunas que você deseja selecionar.

nome_da_tabela: A tabela da qual você deseja recuperar dados.

condição: Uma condição opcional para filtrar os resultados.

Na próxima aula, exploraremos mais a fundo o comando SELECT e aprenderemos a usá-lo para realizar consultas em bancos de dados.

Comandos SQL Básicos

Vamos começar nossa jornada no SQL com alguns comandos básicos. O SQL é uma linguagem de consulta, e os comandos SQL são usados para interagir com bancos de dados relacionais. Alguns comandos que você aprenderá nesta aula incluem:

SELECT: Usado para consultar dados de uma tabela.
FROM: Indica de qual tabela os dados devem ser selecionados.
WHERE: Permite filtrar os resultados com base em condições específicas.

Esses comandos formam a estrutura fundamental para recuperar dados de um banco de dados.

Comando SELECT para Consulta de Dados

O comando SELECT é o coração das consultas SQL. Ele nos permite recuperar dados de uma ou mais tabelas. A sintaxe básica de um comando SELECT é a seguinte:

SELECT column1, column2, ...

FROM table_name

WHERE condition;

column1, column2, ...: As colunas que você deseja selecionar.

table_name: A tabela da qual você deseja recuperar dados.

condition: Uma condição opcional para filtrar os resultados.

Nesta aula, você aprenderá como usar o comando SELECT para recuperar dados específicos de tabelas e como aplicar condições para refinar os resultados.

Cláusulas SELECT (FROM, WHERE, ORDER BY) e Prática

Cláusulas SELECT: FROM, WHERE, ORDER BY

Nesta primeira aula prática de SQL, vamos explorar as cláusulas fundamentais do comando SELECT: FROM, WHERE e ORDER BY. Estas cláusulas são a espinha dorsal de qualquer consulta SQL e são essenciais para recuperar e organizar dados de um banco de dados.

Cláusula FROM

A cláusula FROM é usada para especificar de qual tabela ou tabelas você deseja recuperar dados. Ela é a base de qualquer consulta e define a fonte dos dados. Aqui está um exemplo simples:

```
SELECT *
FROM funcionarios;
```

Neste exemplo, estamos selecionando todas as colunas da tabela "funcionarios". É importante notar que você pode especificar várias tabelas na cláusula FROM se desejar unir dados de diferentes fontes.

Cláusula WHERE

A cláusula WHERE é usada para filtrar os resultados da consulta com base em condições específicas. Ela permite que você extraia apenas os registros que atendem a determinados critérios. Por exemplo:

```
SELECT *
FROM pedidos
WHERE valor_total > 1000;
```

Neste caso, estamos selecionando todos os pedidos cujo "valor_total" seja superior a 1000. A cláusula WHERE é essencial para refinar seus resultados e obter apenas os dados relevantes.

Cláusula ORDER BY

A cláusula ORDER BY é usada para classificar os resultados da consulta em uma ordem específica. Você pode ordenar os resultados em ordem crescente (ASC) ou decrescente (DESC) com base em uma ou mais colunas. Por exemplo:

SELECT nome, salario

FROM funcionarios

ORDER BY salario DESC;

Neste exemplo, estamos selecionando o nome e o salário dos funcionários e ordenando-os em ordem decrescente com base no salário. Isso ajuda a identificar rapidamente os funcionários mais bem remunerados.

Prática: Escrever Consultas SELECT Simples em um Banco de Dados de Exemplo

A melhor maneira de aprender SQL é praticando. Utilizaremos um banco de dados de exemplo para realizar algumas consultas simples.

Exemplo de consulta:

-- Selecione todos os clientes da tabela 'clientes' que são de Nova York e ordene-os por ordem alfabética.

SELECT nome, cidade

FROM clientes

WHERE cidade = 'Nova York'

ORDER BY nome;

Agora é sua vez! Experimente escrever consultas SQL simples para recuperar dados de um banco de dados de exemplo. Lembre-se de usar as cláusulas FROM, WHERE e ORDER BY para especificar a fonte dos dados, filtrar os resultados e ordená-los conforme necessário.

Na próxima aula, exploraremos consultas mais complexas, mas dominar estas cláusulas fundamentais é um passo crucial para se tornar um profissional competente em SQL.

Vamos criar um projeto simples usando um banco de dados de exemplo com a criação e população de tabelas, além de consultas SQL simples que podem ser usadas na Aula 1 do plano de ensino.

Passo 1: Criar o Banco de Dados e Tabelas

Primeiro, criaremos um banco de dados chamado "escola" e duas tabelas: "alunos" e "cursos". Vamos supor que estamos gerenciando informações de alunos e cursos em uma escola.

-- Criar o banco de dados

CREATE DATABASE escola;

-- Usar o banco de dados

USE escola;

-- Criar a tabela de alunos

CREATE TABLE alunos (

id INT PRIMARY KEY,

nome VARCHAR(50),

idade INT,

curso_id INT

);

-- Criar a tabela de cursos

CREATE TABLE cursos (

id INT PRIMARY KEY,

nome VARCHAR(50)

);

-- Popular a tabela de cursos

INSERT INTO cursos (id, nome)

VALUES

(1, 'Matemática'),

(2, 'História'),

(3, 'Ciências');

-- Popular a tabela de alunos

INSERT INTO alunos (id, nome, idade, curso_id)

VALUES

(1, 'João', 18, 1),

(2, 'Maria', 17, 2),

(3, 'Pedro', 19, 1),

(4, 'Ana', 16, 3);

Neste passo, criamos um banco de dados "escola" e duas tabelas: "alunos" e "cursos". A tabela "alunos" tem informações sobre os alunos, incluindo nome, idade e o ID do curso em que estão matriculados. A tabela "cursos" contém informações sobre os cursos oferecidos na escola.

Passo 2: Consultas SQL Simples

Agora, vamos realizar algumas consultas SQL simples para extrair informações do banco de dados.

Consulta 1: Selecionar todos os alunos

SELECT *

FROM alunos;

Consulta 2: Selecionar alunos com idade superior a 18 anos

SELECT nome, idade

FROM alunos

WHERE idade > 18;

Consulta 3: Selecionar alunos do curso de Matemática

SELECT nome

FROM alunos

WHERE curso_id = 1;

Consulta 4: Selecionar cursos oferecidos na escola em ordem alfabética

SELECT nome

FROM cursos

ORDER BY nome;

Estas são consultas SQL simples que podem ser usadas na Aula 1 para demonstrar o uso das cláusulas SELECT, FROM, WHERE e ORDER BY. Você pode executar essas consultas em seu sistema de gerenciamento de banco de dados (SGBD) para ver os resultados. Este projeto serve como um exemplo prático para ajudar os alunos a entender os conceitos apresentados na aula.

Conclusão

Nesta primeira aula, estabelecemos uma base sólida para o nosso curso de SQL. Você aprendeu o que é o SQL, por que é importante e como ele se relaciona com bancos de dados relacionais. Além disso, começamos a explorar os comandos SQL básicos, com foco especial no comando SELECT.

Nesta aula, estabelecemos uma base sólida para nosso curso de SQL. Você aprendeu o que é SQL, por que é importante e como ele se relaciona com bancos de dados relacionais. Além disso, começamos a explorar os comandos SQL básicos, com foco especial no comando SELECT.

Na próxima aula, continuaremos nossa jornada pelo mundo do SQL, aprofundando-nos na filtragem de dados e no uso de operadores lógicos para criar

consultas mais avançadas. Portanto, prepare-se para aprofundar seus conhecimentos em SQL e para se tornar um mestre em consultas de banco de dados!

Na próxima aula, continuaremos nossa jornada pelo mundo do SQL, aprofundando-nos na filtragem de dados e no uso de operadores lógicos para criar consultas mais avançadas. Portanto, prepare-se para aprofundar seus conhecimentos em SQL e para se tornar um mestre em consultas de banco de dados!

Aula 2 - Filtragem de Dados e Operadores Lógicos

Revisão da Cláusula WHERE

Nesta segunda aula, iremos aprofundar nossos conhecimentos sobre a cláusula WHERE, que é essencial para a filtragem de dados em consultas SQL. A cláusula WHERE nos permite extrair informações específicas de um conjunto de dados, permitindo que nossas consultas sejam mais precisas e relevantes.

Operadores de Comparação

Para começar, vamos revisar os operadores de comparação que podem ser usados em conjunto com a cláusula WHERE para definir condições em nossas consultas:

= (Igual a): Usado para verificar se dois valores são iguais.

!= ou <> (Diferente de): Verifica se dois valores são diferentes.

< (Menor que): Verifica se um valor é menor que outro.

> (Maior que): Verifica se um valor é maior que outro.

<= (Menor ou igual a): Verifica se um valor é menor ou igual a outro.

>= (Maior ou igual a): Verifica se um valor é maior ou igual a outro.

Operadores Lógicos (AND, OR, NOT)

Além dos operadores de comparação, podemos usar operadores lógicos para criar condições mais complexas em nossas consultas. Os três operadores lógicos mais comuns são:

AND: Retorna verdadeiro se todas as condições forem verdadeiras. É usado para combinar várias condições.

OR: Retorna verdadeiro se pelo menos uma das condições for verdadeira. É usado quando desejamos que pelo menos uma condição seja atendida.

NOT: Inverte o resultado de uma condição. Retorna verdadeiro se a condição for falsa e vice-versa. É usado para negar uma condição.

Exemplos Práticos

Vamos explorar alguns exemplos práticos para entender como esses operadores funcionam:

Exemplo 1: Selecionar alunos com idade entre 18 e 21 anos que estão no curso de Matemática ou Ciências.

```
SELECT nome, idade, curso
FROM alunos
WHERE (idade >= 18 AND idade <= 21) AND (curso = 'Matemática' OR curso = 'Ciências');
```

Exemplo 2: Selecionar todos os cursos, exceto "História".

```
SELECT nome
FROM cursos
WHERE nome != 'História';
```

Exemplo 3: Selecionar alunos que não estão matriculados em nenhum curso.

```
SELECT nome
FROM alunos
WHERE curso IS NULL;
```

Estes são apenas alguns exemplos para demonstrar como os operadores de comparação e lógicos podem ser usados para filtrar dados em consultas SQL. Na próxima aula, continuaremos nossa jornada explorando funções de agregação e agrupamento, o que nos permitirá realizar análises mais avançadas em nossos

dados. Certifique-se de praticar esses conceitos para aprofundar sua compreensão e habilidades em SQL.

Consultas SQL com Várias Condições - Prática Avançada

Nesta aula, avançaremos nossos conhecimentos em SQL, concentrando-nos em consultas com várias condições. Até agora, aprendemos a usar a cláusula WHERE para filtrar dados com base em uma única condição. Agora, vamos explorar como combinar várias condições para realizar filtragens mais complexas e específicas.

Usando Operadores Lógicos

Para realizar consultas com várias condições, precisamos dos operadores lógicos AND, OR e NOT. Esses operadores nos permitem criar expressões condicionais mais sofisticadas. Eis como eles funcionam:

- AND: Retorna verdadeiro apenas se todas as condições forem verdadeiras. É usado quando queremos que várias condições sejam atendidas.
- OR: Retorna verdadeiro se pelo menos uma das condições for verdadeira. É útil quando queremos que pelo menos uma das condições seja satisfeita.
- NOT: Inverte o resultado de uma condição. Ou seja, se uma condição é verdadeira, NOT a tornará falsa e vice-versa.

Exemplos de Consultas com Várias Condições

Agora, vamos examinar alguns exemplos práticos para ilustrar o uso desses operadores em consultas SQL:

Exemplo 1: Selecionar todos os alunos que têm mais de 18 anos e estão matriculados no curso de Matemática ou Ciências.

```
SELECT nome, idade, curso
FROM alunos
WHERE idade > 18 AND (curso = 'Matemática' OR curso = 'Ciências');
```

Neste exemplo, usamos AND para garantir que a idade seja superior a 18 e OR para permitir que o aluno esteja em um dos cursos especificados.

Exemplo 2: Selecionar todos os alunos que não estão matriculados no curso de História.

```
SELECT nome, curso
FROM alunos
WHERE NOT curso = 'História';
```

Usamos NOT para inverter a condição, de forma que apenas os alunos que não estão matriculados em História sejam selecionados.

Exemplo 3: Selecionar todos os cursos que têm pelo menos 10 alunos e não são de História.

```
SELECT nome
FROM cursos
WHERE NOT nome = 'História' AND (
SELECT COUNT(*)
FROM alunos
WHERE alunos.curso_id = cursos.id
) >= 10;
```

Aqui, usamos AND para combinar duas condições: que o curso não seja História (NOT nome = 'História') e que o número de alunos no curso seja pelo menos 10.

Prática: Escrevendo Consultas SELECT com Filtragem Avançada

A melhor maneira de dominar consultas SQL com várias condições é praticar. Use o seu banco de dados de exemplo ou crie um novo conjunto de dados para praticar consultas que envolvam operadores lógicos. Lembre-se de que quanto mais você praticar, mais confiante e habilidoso se tornará em escrever consultas SQL complexas.

Projeto Prático: Consultas SQL com Várias Condições

Objetivo do Projeto:

Neste projeto prático, você irá aplicar os conceitos aprendidos na Aula 2 sobre consultas SQL com várias condições. Vamos criar um banco de dados de exemplo e realizar consultas avançadas usando operadores lógicos (AND, OR, NOT) e operadores de comparação para extrair informações específicas.

Passos do Projeto:

Passo 1: Criação do Banco de Dados

Crie um banco de dados chamado "escola" e defina duas tabelas: "alunos" e "cursos". As tabelas devem ter a seguinte estrutura:

Tabela "alunos"

id (chave primária)

nome (VARCHAR)

idade (INT)

curso_id (INT)

Tabela "cursos"

id (chave primária)

nome (VARCHAR)

Passo 2: Consultas SQL

Agora, crie consultas SQL que atendam aos seguintes critérios:

Consulta 1: Selecionar todos os alunos com idade entre 18 e 21 anos que estão no curso de Matemática ou Ciências.

Consulta 2: Selecionar todos os cursos que não são de Matemática nem de História.

Consulta 3: Selecionar alunos que não estão matriculados no curso de Ciências ou têm menos de 18 anos.

Consulta 4: Selecionar cursos que têm menos de 5 alunos com idade superior a 20 anos.

Lembre-se de usar operadores lógicos (AND, OR, NOT) e operadores de comparação em suas consultas para atender aos critérios especificados.

Passo 3: Teste e Verificação

Execute suas consultas SQL no banco de dados "escola" para verificar se elas retornam os resultados esperados. Certifique-se de que cada consulta atenda aos critérios definidos no projeto.

Passo 4: Documentação

Crie uma documentação que inclua as consultas SQL que você escreveu, os resultados obtidos e uma breve explicação de como cada consulta funciona. Esta documentação ajudará a consolidar o que você aprendeu neste projeto.

Conclusão:

Este projeto prático oferece uma oportunidade prática para aplicar os conceitos de consultas SQL com várias condições, conforme discutido na Aula 2. Ele permitirá que você desenvolva habilidades sólidas de escrita e execução de consultas SQL complexas, preparando-o para lidar com consultas de dados mais avançadas em seu trabalho ou estudos futuros.

Aula 3 - Ordenação, Funções de Agregação e Grupos

Introdução

Nesta terceira aula do nosso curso de SQL, vamos mergulhar em tópicos mais avançados, mas igualmente importantes: ordenação, funções de agregação, agrupamento de resultados e filtragem de grupos. Esses conceitos são fundamentais para a análise de dados e para a obtenção de informações significativas de grandes conjuntos de dados.

Revisão da Cláusula ORDER BY

Começaremos nossa aula com uma revisão da cláusula ORDER BY. Já aprendemos que essa cláusula nos permite classificar os resultados de uma consulta em ordem crescente (ASC) ou decrescente (DESC) com base em uma ou mais colunas. Isso é crucial quando desejamos visualizar ou relatar dados de maneira organizada e compreensível.

Funções de Agregação

Agora, vamos nos aprofundar nas funções de agregação, que são usadas para calcular valores resumidos em conjuntos de dados. As funções de agregação mais comuns incluem:

- SUM: Calcula a soma dos valores em uma coluna.
- COUNT: Conta o número de linhas em um grupo.
- AVG: Calcula a média dos valores em uma coluna.
- MAX: Retorna o valor máximo em uma coluna.
- MIN: Retorna o valor mínimo em uma coluna.

Essas funções nos permitem extrair informações estatísticas e resumidas de nossos dados, o que é fundamental para análises quantitativas.

Agrupamento de Resultados com GROUP BY

Agora que entendemos como usar funções de agregação, veremos como agrupar resultados com a cláusula GROUP BY. O GROUP BY é usado para criar grupos com base nos valores de uma ou mais colunas e aplicar funções de agregação a cada grupo separadamente. Isso é útil quando queremos analisar dados em nível de grupo.

Filtragem de Grupos com HAVING

Além do GROUP BY, a cláusula HAVING é usada para filtrar grupos com base em condições específicas após a agregação ter ocorrido. Isso permite que você selecione apenas grupos que atendam a determinados critérios.

Prática: Realizar Consultas com Funções de Agregação e Agrupamento

A melhor maneira de aprender esses conceitos é praticando. Para a prática, crie consultas SQL que utilizem funções de agregação, agrupamento e filtragem de grupos. Você pode usar um banco de dados de exemplo ou criar um conjunto de dados fictício. Certifique-se de criar consultas que explorem várias funções de agregação, grupos e condições de filtragem.

Conclusão

Nesta aula, exploramos tópicos avançados em SQL que são essenciais para a análise de dados. Aprendemos como usar funções de agregação, agrupamento de resultados e filtragem de grupos para obter informações significativas a partir de nossos dados. Com essas habilidades, você estará preparado para realizar análises mais avançadas e fornecer insights valiosos em suas consultas SQL. Continue praticando para aprofundar seu conhecimento e habilidades nesta área. Na próxima aula, continuaremos nossa jornada explorando subconsultas, que nos permitirão criar consultas SQL ainda mais poderosas.

Aula 4 - Subconsultas Simples

Introdução às Subconsultas

Nesta quarta aula do nosso curso de SQL, entraremos no mundo das subconsultas, uma ferramenta poderosa para realizar consultas mais complexas e obter informações específicas de um banco de dados. As subconsultas, também conhecidas como consultas aninhadas ou consultas internas, permitem que você insira uma consulta dentro de outra consulta SQL.

Subconsultas como Expressões

Em essência, uma subconsulta é uma consulta que é tratada como uma única expressão em uma consulta maior. Você pode pensar nelas como consultas dentro de consultas. As subconsultas podem ser usadas em várias cláusulas SQL, como SELECT, FROM, WHERE, HAVING, e até mesmo INSERT e UPDATE.

Subconsultas em Cláusulas WHERE

Uma das maneiras mais comuns de usar subconsultas é na cláusula WHERE. Isso permite que você filtre os resultados com base em valores retornados por uma consulta interna. Por exemplo:

```
SELECT nome
FROM alunos
WHERE idade > (SELECT AVG(idade) FROM alunos);
```

Neste exemplo, a subconsulta (SELECT AVG(idade) FROM alunos) calcula a idade média dos alunos e a compara com a idade de cada aluno na tabela. Apenas os alunos com idade superior à média serão retornados.

Subconsultas em Cláusulas FROM

Você também pode usar subconsultas na cláusula FROM para criar temporariamente uma tabela virtual que pode ser usada na consulta principal. Por exemplo:

```
SELECT a.nome, c.nome as curso
FROM (SELECT * FROM alunos WHERE idade > 18) a
JOIN cursos c ON a.curso_id = c.id;
```

Neste caso, a subconsulta (SELECT * FROM alunos WHERE idade > 18) cria uma tabela temporária a contendo apenas os alunos com mais de 18 anos. Isso permite que você realize uma junção com a tabela de cursos e obtenha os nomes dos cursos frequentados por esses alunos.

Prática: Criar Consultas com Subconsultas Simples

A melhor maneira de aprender sobre subconsultas é praticar. Crie consultas SQL que envolvam subconsultas simples em seu banco de dados de exemplo ou em um conjunto de dados fictício. Experimente criar consultas que usem subconsultas em cláusulas WHERE e FROM, e veja como as subconsultas podem ser usadas para filtrar e transformar seus resultados de maneira mais eficaz.

As subconsultas são uma ferramenta valiosa em SQL e podem tornar suas consultas mais flexíveis e poderosas. Continue praticando e explorando as possibilidades das subconsultas à medida que avançamos em nosso curso. Na próxima aula, exploraremos subconsultas correlacionadas e subconsultas aninhadas, que levam essa técnica a um nível ainda mais avançado.

Aula 5 - Subconsultas Correlacionadas e Subconsultas Aninhadas

Introdução

Bem-vindos à quinta aula do nosso curso de SQL avançado. Nesta aula, exploraremos um nível mais profundo de subconsultas: as subconsultas correlacionadas e as subconsultas aninhadas. Essas técnicas nos permitem realizar consultas ainda mais sofisticadas e resolver problemas complexos em bancos de dados.

Subconsultas Correlacionadas

Uma subconsulta correlacionada é uma subconsulta que faz referência a colunas da consulta externa. Em outras palavras, os resultados da subconsulta são dependentes dos valores da consulta principal. Isso permite que você realize consultas mais contextuais e específicas.

Exemplo de Subconsulta Correlacionada:

```
SELECT nome
FROM alunos a
WHERE idade > (SELECT AVG(idade) FROM alunos WHERE curso_id = a.curso_id);
```

Neste exemplo, a subconsulta (SELECT AVG(idade) FROM alunos WHERE curso_id = a.curso_id) é correlacionada porque faz referência à coluna curso_id da consulta externa. Ela calcula a idade média dos alunos no mesmo curso que o aluno da consulta principal. Isso nos permite selecionar apenas os alunos cuja idade é superior à média de idade de seu próprio curso.

Subconsultas Aninhadas

As subconsultas aninhadas, como o nome sugere, são subconsultas dentro de subconsultas. Elas podem ser usadas para criar consultas altamente complexas e resolver problemas que envolvem múltiplos níveis de aninhamento.

Exemplo de Subconsulta Aninhada:

```
SELECT nome
FROM cursos
WHERE id IN (SELECT curso_id FROM alunos WHERE idade > 18);
```

Neste exemplo, a subconsulta interna (SELECT curso_id FROM alunos WHERE idade > 18) retorna uma lista de IDs de cursos frequentados por alunos com mais de 18 anos. A consulta externa, em seguida, seleciona os nomes dos cursos com base nessa lista de IDs.

Uso de Subconsultas para Resolver Problemas Complexos

Subconsultas correlacionadas e aninhadas são frequentemente usadas para resolver problemas complexos que envolvem consultas de dados interdependentes. Elas permitem que você quebre problemas em partes menores e, em seguida, os resolva passo a passo.

Prática: Escrever Consultas Avançadas com Subconsultas Correlacionadas e Aninhadas

A melhor maneira de dominar subconsultas correlacionadas e aninhadas é praticar. Crie consultas SQL que usem essas técnicas em seu banco de dados de exemplo ou em um conjunto de dados fictício. Explore casos de uso onde a dependência entre as consultas é necessária para obter os resultados desejados.

À medida que você se torna mais proficiente com subconsultas correlacionadas e aninhadas, estará preparado para resolver problemas complexos de análise de dados em seu trabalho ou estudos futuros.

Conclusão

Nesta aula, exploramos as subconsultas correlacionadas e aninhadas, técnicas avançadas que podem ser aplicadas para resolver problemas complexos em SQL. Com essas habilidades, você pode enfrentar consultas de dados mais desafiadoras e se tornar um especialista em análise de dados. Continue praticando e explorando os limites das subconsultas em seu aprendizado contínuo de SQL.

Projeto: Gestão de Biblioteca com MySQL

Descrição do Projeto:

Neste projeto prático, você construirá um sistema de gerenciamento de biblioteca usando MySQL. Este projeto envolve a criação de um banco de dados que rastreia informações sobre livros, autores, empréstimos e usuários da biblioteca. Você utilizará conceitos aprendidos nas aulas 1 a 5 para criar e consultar este banco de dados.

Estrutura do Banco de Dados:

O banco de dados da biblioteca será composto por várias tabelas:

livros: Armazena informações sobre os livros, incluindo título, ISBN e autor.
autores: Contém detalhes sobre os autores, como nome e nacionalidade.
usuários: Registra informações sobre os usuários da biblioteca, como nome, número de telefone e endereço de e-mail.
empréstimos: Rastreia os empréstimos de livros, incluindo a data de início, a data de retorno prevista e o usuário que fez o empréstimo.

Passos do Projeto:

Passo 1: Criação do Banco de Dados

- Crie um banco de dados chamado "biblioteca" no MySQL.

Passo 2: Criação das Tabelas

- Crie as tabelas "livros", "autores", "usuários" e "empréstimos" com as colunas apropriadas.

Passo 3: Inserção de Dados

- Popule as tabelas com dados fictícios de livros, autores, usuários e empréstimos. Certifique-se de que haja exemplos suficientes para permitir consultas significativas.

Passo 4: Consultas SQL

Agora, escreva consultas SQL para realizar as seguintes operações:

Aula 1 - Introdução ao SQL e SELECT:

- Consultar todos os livros disponíveis na biblioteca.

Aula 2 - Filtragem de Dados e Operadores Lógicos:

- Selecionar todos os livros emprestados atualmente.
- Encontrar todos os livros em atraso (data de retorno prevista expirada).

Aula 3 - Ordenação, Funções de Agregação e Grupos:

- Encontrar o autor com mais livros na biblioteca.
- Calcular o número total de livros por autor.

Aula 4 - Subconsultas Simples:

- Encontrar todos os usuários que têm pelo menos um livro emprestado.
- Selecionar todos os livros emprestados por um usuário específico.

Aula 5 - Subconsultas Correlacionadas e Subconsultas Aninhadas:

- Determinar o usuário que mais emprestou livros.
- Encontrar os livros mais emprestados.

Passo 5: Documentação

Documente suas consultas SQL e os resultados obtidos. Explicite o propósito de cada consulta, como ela foi construída e os resultados esperados.

Conclusão:

Este projeto de gerenciamento de biblioteca com MySQL é uma ótima maneira de aplicar os conceitos aprendidos no curso. Além de aprimorar suas habilidades em SQL, você também desenvolverá um projeto funcional que pode ser usado como base para sistemas de bibliotecas reais. Lembre-se de que a prática constante é fundamental para se tornar um profissional habilidoso em SQL, e projetos como este oferecem uma oportunidade prática para fazer isso.

Para o projeto de gerenciamento de biblioteca com MySQL, você precisará de várias cláusulas SQL para realizar consultas e operações no banco de dados. Aqui estão algumas das cláusulas SQL que você usaria comumente para este projeto:

Criação do Banco de Dados

```
CREATE DATABASE biblioteca;
```

Selecionar um Banco de Dados:

```
USE biblioteca;
```

Criação de Tabelas:

```
CREATE TABLE livros (
id INT AUTO_INCREMENT PRIMARY KEY,
titulo VARCHAR(255) NOT NULL,
isbn VARCHAR(13) NOT NULL,
autor_id INT,
disponivel BOOLEAN DEFAULT TRUE
```

```
);
CREATE TABLE autores (
id INT AUTO_INCREMENT PRIMARY KEY,
nome VARCHAR(255) NOT NULL,
nacionalidade VARCHAR(100)
);
CREATE TABLE usuarios (
id INT AUTO_INCREMENT PRIMARY KEY,
nome VARCHAR(255) NOT NULL,
telefone VARCHAR(15),
email VARCHAR(100)
);
CREATE TABLE emprestimos (
id INT AUTO_INCREMENT PRIMARY KEY,
livro_id INT,
usuario_id INT,
data_emprestimo DATE,
data_devolucao DATE
);
```

Inserção de Dados:

INSERT INTO autores (nome, nacionalidade) VALUES ('Autor 1', 'Nacionalidade 1');

INSERT INTO livros (titulo, isbn, autor_id, disponivel) VALUES ('Livro 1', '1234567890123', 1, TRUE);

INSERT INTO usuarios (nome, telefone, email) VALUES ('Usuário 1', '123-456-7890', 'usuario1@example.com');

INSERT INTO emprestimos (livro_id, usuario_id, data_emprestimo, data_devolucao) VALUES (1, 1, '2023-01-01', '2023-01-15');

Consultas SQL:

-- Consultar todos os livros disponíveis na biblioteca

SELECT * FROM livros WHERE disponivel = TRUE;

-- Selecionar todos os livros emprestados atualmente

SELECT * FROM livros WHERE disponivel = FALSE;

-- Encontrar todos os livros em atraso (data de retorno prevista expirada)

SELECT * FROM emprestimos WHERE data_devolucao < CURDATE();

-- Encontrar o autor com mais livros na biblioteca

SELECT autores.nome AS autor, COUNT(livros.id) AS quantidade_de_livros

FROM autores

```
JOIN livros ON autores.id = livros.autor_id

GROUP BY autores.id

ORDER BY quantidade_de_livros DESC

LIMIT 1;


-- Calcular o número total de livros por autor

SELECT autores.nome AS autor, COUNT(livros.id) AS quantidade_de_livros

FROM autores

JOIN livros ON autores.id = livros.autor_id

GROUP BY autores.id;


-- Encontrar todos os usuários que têm pelo menos um livro emprestado

SELECT usuarios.nome AS usuario

FROM usuarios

JOIN emprestimos ON usuarios.id = emprestimos.usuario_id;


-- Selecionar todos os livros emprestados por um usuário específico

SELECT livros.titulo AS livro, emprestimos.data_emprestimo,
emprestimos.data_devolucao

FROM emprestimos

JOIN livros ON emprestimos.livro_id = livros.id

WHERE emprestimos.usuario_id = 1;
```

```
-- Determinar o usuário que mais emprestou livros

SELECT usuarios.nome AS usuario, COUNT(emprestimos.id) AS
quantidade_de_emprestimos

FROM usuarios

JOIN emprestimos ON usuarios.id = emprestimos.usuario_id

GROUP BY usuarios.id

ORDER BY quantidade_de_emprestimos DESC

LIMIT 1;


-- Encontrar os livros mais emprestados

SELECT livros.titulo AS livro, COUNT(emprestimos.id) AS
quantidade_de_emprestimos

FROM livros

JOIN emprestimos ON livros.id = emprestimos.livro_id

GROUP BY livros.id

ORDER BY quantidade_de_emprestimos DESC;
```

Essas são algumas das cláusulas SQL que você pode usar para criar e consultar o banco de dados de gerenciamento de biblioteca. Lembre-se de adaptar as consultas às suas necessidades específicas à medida que trabalha no projeto.

Pensar Letras, Sentir Palavras