

Banco de  
Dados  
Com MySQL



WOMAKERS<sup>®</sup>  
CODE

# Sub-Queries

## Definição

São instruções ou comandos **SELECT** dentro de outra instrução.

```
SELECT *  
FROM city  
WHERE country_id = (  
    SELECT country_id  
    FROM country  
    WHERE country = "Brazil"  
    LIMIT 1  
);
```

# Sub-Queries

## Definição

- ❑ Podem ser utilizadas em qualquer lugar que permita uma expressão.
- ❑ Também conhecidas como consultas internas ou subconsultas.
- ❑ Um **SELECT** que funciona como subconsulta deve sempre ser colocado entre parênteses.
- ❑ Pode ficar na cláusula **WHERE** ou **HAVING** e ser usada com **SELECT**, **DELETE**, **INSERT** e **UPDATE**.

# Sub-Queries

## Exemplos

Funcionam como listas no **IN** ou com operações de comparação como **ANY** ou **ALL**:

```
SELECT *  
FROM address  
WHERE city_id IN (  
    SELECT city_id  
    FROM city  
    WHERE country_id = 15  
);
```

# Sub-Queries

## Exemplos

Utilizadas com operadores de comparação = e neste caso devem retornar apenas um valor:

```
SELECT *  
FROM city  
WHERE country_id = (  
    SELECT country_id  
    FROM country  
    WHERE country = "Brazil"  
    LIMIT 1  
);
```

# Sub-Queries

## Exemplos

- ❏ Testes de existência utilizados com **EXISTS**:

```
SELECT *  
FROM film  
WHERE EXISTS (  
    SELECT film_id  
    FROM film_inventory  
);
```

# Sub-Queries

## Exercícios

1. Selecione nome e sobrenome de todos os funcionários de escritórios nos Estados Unidos (USA).
2. Selecione número e pagamento do funcionário que tem o maior pagamento. Dica: Utilize a função MAX() na subconsulta.
3. Selecione os funcionários que têm pagamento acima da média. Dica: utilize a função AVG() na subconsulta.

# LIMIT

Limitar resultados das consultas

Nem sempre vamos querer que nossas consultas retornem o total de resultados que existem em uma tabela.

**Os motivos para isso podem ser** dos mais variados, sendo os principais: **performance** e **usabilidade**.

**Performance:** uma tabela pode ser extremamente grande em termos de dados armazenados e por isso pode demorar muito para retornar os dados pedidos.

**Usabilidade:** ao colocar muitos dados para o usuário visualizar, pode ficar um pouco confuso. Uma solução é usar o **LIMIT** para retornar apenas uma porção de cada vez.



# LIMIT

## Exemplos

Um exemplo bem simples de como usar o limite seria:

```
SELECT *  
FROM sakila.film  
LIMIT 3;
```

No caso acima seria retornado apenas 3 filmes, não importando quantos de fato tiverem na base de dados.

# LIMIT

## Exemplos

Observe que o comando **LIMIT** pode ser combinado com outros comandos que já vimos, como o **WHERE**. Exemplo:

```
SELECT *  
FROM sakila.film  
WHERE title LIKE "%STONE%"  
LIMIT 3;
```

Nesse exemplo veremos uma lista filtrada pelo título, porém não mais do que 3.

# LIMIT

Usando o offset

O comando **LIMIT** também tem um parâmetro opcional que pode ser usado para definir de onde a lista irá começar. O seu nome é **offset**, e quando omitido, ele é equivalente a zero.

Esses dois exemplos têm exatamente o mesmo efeito:

```
SELECT *  
FROM sakila.film  
LIMIT 3;
```

```
SELECT *  
FROM sakila.film  
LIMIT 0, 3;
```

# LIMIT

Usando o offset

Um exemplo simples, partindo do nosso exemplo anterior, seria:

```
SELECT *  
FROM sakila.film  
LIMIT 5, 3;
```

Observem que agora temos um "5, 3" em vez do "3" que tínhamos antes.

O **5** é de onde os resultados começaram a ser exibidos.

O **3** é a quantidade de resultados retornados

# LIMIT

Exemplo de paginação

Se o **tamanho de cada página é 3**, então:

*# Página 1*

```
SELECT name  
FROM category  
LIMIT 3;
```

Resultado: Action, Animation, Children

Posição	name
0	Action
1	Animation
2	Children
3	Classics
4	Comedy
5	Documentary
6	Drama
7	Family
8	Foreign
9	Games
...	...

# LIMIT

Exemplo de paginação

Se o **tamanho de cada página é 3**, então:

*# Página 2*

```
SELECT name  
FROM category  
LIMIT 3, 3;
```

Resultado: Classics, Comedy, Documentary

Posição	name
0	Action
1	Animation
2	Children
3	Classics
4	Comedy
5	Documentary
6	Drama
7	Family
8	Foreign
9	Games
...	...

# LIMIT

Exemplo de paginação

Se o **tamanho de cada página é 3**, então:

*# Página 3*

```
SELECT name  
FROM category  
LIMIT 6, 3;
```

Resultado: Drama, Family, Foreign

Posição	name
0	Action
1	Animation
2	Children
3	Classics
4	Comedy
5	Documentary
6	Drama
7	Family
8	Foreign
9	Games
...	...



# LIMIT

## Exercícios

1. Busque uma lista de atores, porém com apenas 5 elementos nela.
2. Faça uma query que liste 5 filmes que começam com a letra J.
3. Faça uma lista de três filmes que começam com a letra A, mas que estejam na posição 5 ou acima.



# HAVING

## Definição

O **HAVING** é uma cláusula SQL de comportamento quase idêntico ao **WHERE**, ambos tem o objetivo de filtrar a lista de resultados dada uma determinada condição.

A diferença vem pelo fato de o **HAVING** poder ser utilizado junto com funções agregadoras e mesmo resultados das queries após agrupamentos, enquanto o **WHERE** só pode ser utilizado em valores que já estão escritos na base de dados.

A cláusula **HAVING** pode ser aplicada junto a **WHERE**, porém a **WHERE** será o primeiro filtro a ser executado sobre o resultado.

# HAVING

## Exemplos

Caso a gente queria ver a quantidade de atores que já atuaram em mais de 2 filmes:

```
SELECT  
    CONCAT(a.first_name, ' ', a.last_name),  
    COUNT(f.film_id) total  
FROM sakila.actor a  
JOIN sakila.film_actor fa ON a.actor_id = fa.actor_id  
JOIN sakila.film f ON f.film_id = fa.film_id  
GROUP BY a.actor_id  
HAVING total > 2;
```

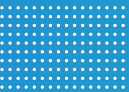
Neste exemplo, teríamos retornado todos os atores que tiveram atuação em mais de dois filmes e também suas respectivas quantidades.

Observe que neste caso, não seria possível usar o **WHERE** para este filtro, já que no momento da execução do **WHERE**, o resultado do **SUM** ainda não foi processado.

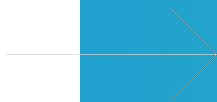
# HAVING

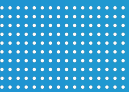
## Exercícios

1. Retorne a lista de atores que já atuaram em mais de 3 dramas.
2. Liste os filmes que têm registrado mais de 8 atores.
3. Liste as linguagens que têm mais de 5 filmes.



**Thank you!**





Evolution is our core

