

Banco de  
Dados  
Com MySQL



WOMAKERS<sup>®</sup>  
CODE

# FUNÇÃO

## Definição

O MySQL tem diversas funções já disponíveis que podem simplesmente serem usadas conforme o necessário.

Um exemplo de função é o **CONCAT**, que pode ser utilizado para unir resultados de diversos campos diferentes em um só.

Exemplo:

```
SELECT CONCAT(first_name, last_name) FROM sakila.actor;
```

# FUNÇÃO

## Lista de funções

A lista de algumas funções úteis do MySQL:

**UPPER:** Coloca todas as letras em maiúsculas

**LOWER:** Transforma o texto em minúsculas

**CONCAT:** Concatena em um campo pedaço de texto e/ou outros campos

**EXTRACT:** Extrai uma unidade de tempo de um timestamp

**REPLACE:** Troca um texto por algum outro

**FORMAT:** Exibe um número com a quantidade de casa decimais desejada

**SUM:** Soma um conjunto de valores

**COUNT:** Retorna o número de registros retornado por um SELECT

Veja a lista completa de **funções e como utilizar** no [W3Schools](https://www.w3schools.com/mysql/func_mysql.asp).

# FUNÇÃO

## Combinando funções

Digamos que fosse necessário buscar a lista de nome completa de atores, mas em apenas um campo e com somente as primeiras letras em maiúsculo.

Neste caso, vamos combinar as funções *CONCAT*, *SUBSTRING* e *LOWER*.

### SELECT

```
CONCAT(  
    SUBSTRING(first_name, 1, 1),  
    LOWER(SUBSTRING(first_name, 2)),  
    '  
    SUBSTRING(last_name, 1, 1),  
    LOWER(SUBSTRING(last_name, 2))  
) AS 'Nome Completo'
```

### FROM

```
sakila.actor;
```

Result Grid		Filter Rows:	Search	Export:
	Nome Completo			
	Penelope Guinness			
	Nick Wahlberg			
	Ed Chase			
	Jennifer Davis			
	Johnny Lollobrigida			
	Bette Nicholson			
	Grace Mostel			
	Matthew Johansson			
	Joe Swank			
	Christian Gable			
	Zero Cage			
	Karl Berry			
	Uma Wood			
	Vivien Bergen			
	Cuba Olivier			

# FUNÇÃO

## Exercícios

- 1 - Na tabela de atores, separe em dia, mês e ano o campo last\_update.
- 2 - Some todos os replacement\_cost na tabela film.

# GROUP BY

## Definição

Este comando agrupa as linhas retornadas de acordo com um campo definido, sempre sendo usado com alguma função agregadora como o **COUNT** e o **SUM**.

Assim, é possível agrupar o resultado pelo valor desejado.

Como por exemplo:


### SELECT

district, **COUNT**(address\_id) total

### FROM

sakila.address

### GROUP BY district

Result Grid			Filter Rows: <input type="text" value="Search"/>	Export: 
	district	total		
	Buenos Aires	10		
	California	9		
	West Bengali	9		
	Shandong	9		
	Uttar Pradesh	8		
	São Paulo	8		
	England	7		
	Maharashtra	7		
	Southern Tagalog	6		
	Texas	5		
	Karnataka	5		
	Gois	5		

# FUNÇÕES DE AGREGAÇÃO

Utilizando funções que agrupam dados junto com **GROUP BY**

Algumas funções como o **COUNT()** e **SUM()** fazem parte de um grupo de funções chamado **Aggregate Functions** (Funções de Agregação), essas funções realizam uma operação em múltiplas linhas e retornam um resultado.

Nós precisamos utilizar uma cláusula **GROUP BY** no final da nossa query para dizer em quais grupos essas operações devem ser executadas.

Por exemplo, se utilizarmos um **COUNT()** precisamos indicar ao MySQL **o quê ele deve contar**. Se utilizarmos um **GROUP BY [name]**, ele vai então contar todos os nomes iguais e dar um retorno. No exemplo abaixo, o nome “Womakers” aparece 17 vezes na tabela.

name	count()
Womakers	17

# GROUP BY

## Exemplo

Vamos agora buscar a quantidade de filmes por classificação.  
Veja como fica query:

```
SELECT
    rating, COUNT(film_id) total
FROM
    sakila.film
GROUP BY rating
```

Result Grid			Filter Rows:	Search	Export:
	rating	total			
▶	PG	194			
◀	G	178			
	NC-17	210			
◀	PG-13	223			
	R	195			



# GROUP BY

## Exercicio

- 1 - Na tabela actor, agrupe os registros pelo last\_name para saber quantos atores têm o mesmo sobrenome.
- 2- Tente utilizar uma função de **COUNT()** sem utilizar o **GROUP BY** e veja o que acontece

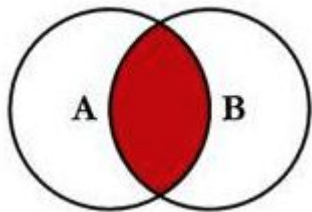
# INNER JOIN

## Definição

Os dados de nossas tabelas raramente estão completos por si só. Geralmente, precisamos buscar dados de diversas tabelas para atingirmos nossos objetivos.

Existem diversas variações, a mais sendo o **INNER JOIN**. Esta variação irá retornar todos os registros da Tabela A e da Tabela B quando estes tiverem correspondente uma na outra.

Exemplo:



```
SELECT * FROM sakila.film f JOIN sakila.category c ON f.category_id = c.category_id
```

# INNER JOIN

## Exemplo

Vamos unir os resultados das tabelas film e category:

**SELECT**

\*

**FROM**

sakila.film f **JOIN** sakila.category c  
**ON** f.category\_id = c.category\_id

**OBS:** Note que as colunas das duas tabelas foram recuperadas.

rating	special_features	last_update	language_id	name	last_update
PG	Deleted Scenes,Behind the Scenes	2006-02-15 05:03:42	1	English	2006-02-15 05:0
G	Trailers,Deleted Scenes	2006-02-15 05:03:42	1	English	2006-02-15 05:0
NC-17	Trailers,Deleted Scenes	2006-02-15 05:03:42	1	English	2006-02-15 05:0
G	Commentaries,Behind the Scenes	2006-02-15 05:03:42	1	English	2006-02-15 05:0
G	Deleted Scenes	2006-02-15 05:03:42	1	English	2006-02-15 05:0
PG	Deleted Scenes	2006-02-15 05:03:42	1	English	2006-02-15 05:0
PG-13	Trailers,Deleted Scenes	2006-02-15 05:03:42	1	English	2006-02-15 05:0
R	Trailers	2006-02-15 05:03:42	1	English	2006-02-15 05:0
PG-13	Trailers,Deleted Scenes	2006-02-15 05:03:42	1	English	2006-02-15 05:0
NC-17	Trailers,Deleted Scenes	2006-02-15 05:03:42	1	English	2006-02-15 05:0
G	Commentaries,Behind the Scenes	2006-02-15 05:03:42	1	English	2006-02-15 05:0
PG	Commentaries,Deleted Scenes	2006-02-15 05:03:42	1	English	2006-02-15 05:0
PG	Deleted Scenes,Behind the Scenes	2006-02-15 05:03:42	1	English	2006-02-15 05:0

# INNER JOIN

## Exemplo

Podemos especificar quais campos queremos das duas tabelas do join:

### SELECT

f.title, l.name

### FROM

sakila.film f **JOIN** sakila.category c

**ON** f.category\_id = c.category\_id

**OBS:** Note podemos dar apelidos para as tabelas assim como damos para os campos.  
Neste caso, f para film e l para language.

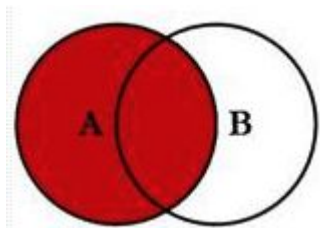
Result Grid			Filter Rows:	Search	Export:	Fetch rows:
title	name					
▶ ACADEMY DINOSAUR	English					
▶ ACE GOLDFINGER	English					
▶ ADAPTATION HOLES	English					
▶ AFFAIR PREJUDICE	English					
▶ AFRICAN EGG	English					
▶ AGENT TRUMAN	English					
▶ AIRPLANE SIERRA	English					
▶ AIRPORT POLLOCK	English					
▶ ALABAMA DEVIL	English					
▶ ALADDIN CALENDAR	English					
▶ ALAMO VIDEOTAPE	English					
▶ ALASKA PHANTOM	English					
▶ ALL FOREVER	English					

# LEFT JOIN

## Definição

Nesta variação do join, a tabela da esquerda (tabela film) vai ter seus dados retornados mesmo que não tenha uma correspondência na tabela da direita. Exemplo:

```
SELECT f.title, l.name FROM sakila.film f LEFT JOIN sakila.language l ON f.language_id = l.language_id
```



**OBS:** O primeiro filme não tem a língua definida, mas o registro foi recuperado mesmo assim.

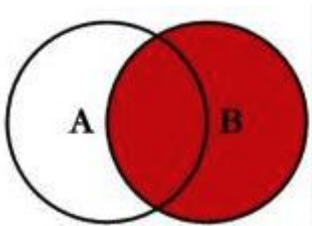
Result Grid			Filter Rows:	Search	Export:	Fetch rows:
title	name					
▶ ACADEMY DINOSAUR	NULL					
▶ ACE GOLDFINGER	English					
▶ ADAPTATION HOLES	English					
▶ AFFAIR PREJUDICE	English					
▶ AFRICAN EGG	English					
▶ AGENT TRUMAN	English					
▶ AIRPLANE SIERRA	English					
▶ AIRPORT POLLOCK	English					
▶ ALABAMA DEVIL	English					
▶ ALADDIN CALENDAR	English					
▶ ALAMO VIDEOTAPE	English					
▶ ALASKA PHANTOM	English					
▶ ALL FOREVER	English					

# RIGHT JOIN

## Definição

Esta variação traz os dados do lado direito (tabela language), mesmo que não existe correspondência.

```
SELECT f.title, l.name FROM sakila.film f RIGHT JOIN sakila.language l ON f.language_id = l.language_id
```



**OBS:** Desta vez, a primeira língua que não tem correspondência em filme, Mas também foi recuperada.

Result Grid			Filter Rows:	Search	Export:	Fetch rows:
	title	name				
▶	NULL	Italian				
▶	ACE GOLDFINGER	English				
▶	ADAPTATION HOLES	English				
▶	AFFAIR PREJUDICE	English				
▶	AFRICAN EGG	English				
▶	AGENT TRUMAN	English				
▶	AIRPLANE SIERRA	English				
▶	AIRPORT POLLOCK	English				
▶	ALABAMA DEVIL	English				
▶	ALADDIN CALENDAR	English				
▶	ALAMO VIDEOTAPE	English				
▶	ALASKA PHANTOM	English				
▶	ALL I EVER	English				

# LEFT OU RIGHT JOIN?

Como eu sei quando devo usar o **LEFT JOIN** ou **RIGHT JOIN**

Depende de qual tabela você quer que retorne os dados independente de ter registros correspondentes ou não

A referência de “esquerda” ou “direita” é feita com base na posição das tabelas em relação ao **JOIN**

```
SELECT * FROM tabela_esquerda JOIN tabela_direita
```

# JOINS

## Exercicios

1 - Faça um select que recupere o nome do cliente na tabela customer e o distrito na tabela address.

2 - Busca a quantidade de filmes agrupando pelo idioma do filme.

3 - Recupere o nome e sobrenome do cliente (customer) e a quantidade de locações (rental) que ele fez, com ordenação do maior para o menor.

**Desafio:** descobrir qual o filme mais lucrativo para a locadora até o momento.



## Encerramento Aula 3

Críticas, dúvidas, sugestões?



Evolution is our core

