

MySQL



Especialização em Queries

2

1. Funções de agregação.
2. Funções típicas.
3. Group by.
4. Having.
5. Distinct.
6. Sub-queries.
7. Exists.

Agrupamento de dados

3

Problemas a resolver:

- Quantos filmes eu tenho?
- Quantos filmes eu tenho do gênero X?
- Qual é a classificação média dos filmes?
- Qual é a classificação mínima/máxima de um filme?

Funções de agregação

4

- COUNT
- MIN
- MAX
- SUM
- AVG

COUNT

5

```
SELECT COUNT(*)  
FROM movies;
```

```
SELECT COUNT(id)  
FROM movies;
```

```
SELECT COUNT(id) AS total  
FROM movies  
WHERE genre_id = 3;
```

AVG, SUM, MIN, MAX

6

```
SELECT AVG(rating)
FROM movies;
```

```
SELECT SUM(length)
FROM movies;
```

```
SELECT MIN(rating)
FROM movies;
```

```
SELECT MAX(rating)
FROM movies;
```

Mais funções interessantes

7

- CONCAT
- COALESCE
- DATEDIFF
- EXTRACT
- LENGTH
- REPLACE
- DATE FORMAT
- CASE
- DISTINCT

<https://dev.mysql.com/doc/refman/5.7/en/func-op-summary-ref.html>

CONCAT

8

```
SELECT CONCAT('Hola ', 'a ', 'todos ', 'en ', 'Digital ', 'House!');
```

Resultado: Hola a todos en Digital House!

```
SELECT CONCAT('La respuesta es: ', 24);
```

Resultado: La respuesta es: 24

```
SELECT CONCAT('Uniendo dos campos: ', first_name, ' ', last_name)  
FROM actors;
```

Resultado: Uniendo dos campos: Billy Zane

```
SELECT CONCAT('Uniendo con NULL: ', NULL);
```

Resultado: NULL

COALESCE

9

TABLE STRUCTURE

id	customername	mobile	home	work
1	Joe	123	456	789
2	Jane		654	987
3	John			321

```
SELECT id, customername, COALESCE(mobile, home, work) AS phone FROM customers
```

RESULT

id	customername	phone
1	Joe	123
2	Jane	654
3	John	321

DATEDIFF

10

```
mysql> SELECT DATEDIFF('2014-01-28', '2014-01-27');
```

```
Result: 1
```

```
mysql> SELECT DATEDIFF('2014-01-28 11:41:14', '2014-01-27 12:10:08');
```

```
Result: 1
```

```
mysql> SELECT DATEDIFF('2014-01-28 11:41:14', '2014-01-27');
```

```
Result: 1
```

```
mysql> SELECT DATEDIFF('2014-02-15', '2014-02-10');
```

```
Result: 5
```

```
mysql> SELECT DATEDIFF('2014-01-28', '2013-12-31');
```

```
Result: 28
```

```
mysql> SELECT DATEDIFF('2013-12-31', '2014-01-28');
```

```
Result: -28
```

EXTRACT

11

```
mysql> SELECT EXTRACT(SECOND FROM '2014-02-13 08:44:21');  
Result: 21
```

```
mysql> SELECT EXTRACT(MINUTE FROM '2014-02-13 08:44:21');  
Result: 44
```

```
mysql> SELECT EXTRACT(HOUR FROM '2014-02-13 08:44:21');  
Result: 8
```

```
mysql> SELECT EXTRACT(DAY FROM '2014-02-13');  
Result: 13
```

EXTRACT

12

```
mysql> SELECT EXTRACT(WEEK FROM '2014-02-13');
```

```
Result: 6
```

```
mysql> SELECT EXTRACT(MONTH FROM '2014-02-13');
```

```
Result: 2
```

```
mysql> SELECT EXTRACT(QUARTER FROM '2014-02-13');
```

```
Result: 1
```

```
mysql> SELECT EXTRACT(YEAR FROM '2014-02-13');
```

```
Result: 2014
```

REPLACE

13

```
SELECT id,  
REPLACE(title, 'Harry', 'Pedro') AS titulo_manipulado  
FROM movies  
ORDER BY id;
```

```
SELECT REPLACE('abc abc', 'a', 'B');
```

Resultado: Bbc Bbc

```
SELECT REPLACE('abc abc', 'A', 'B');
```

Resultado: abc abc

```
SELECT REPLACE('123 123', '2', '5');
```

Resultado: 153 153

DATE FORMAT

14

```
3 • SELECT
4     id,
5     title,
6     rating,
7     release_date,
8     DATE_FORMAT(release_date, '%W %M %Y') AS fecha_de_estreno
9 FROM movies
10 ORDER BY rating
11
```

Result Grid   Filter Rows: <input type="text"/> 							Export:  Wrap Cell Content: 	
#	id	title	rating	release_date	fecha_de_estreno			
1	7	Transformers: el lado oscuro de la l...	0.9	2005-07-04 00:00:00	Monday July 2005			
2	14	Toy Story 2	3.2	2003-04-04 00:00:00	Friday April 2003			
3	16	Mi pobre angelito	3.2	1989-01-04 00:00:00	Wednesday January 1989			

CASE

15

```
SELECT
  id,
  title,
  rating,
  CASE
    WHEN rating < 4 THEN 'Mala'
    WHEN rating < 6 THEN 'Regular'
    WHEN rating < 8 THEN 'Buena'
    WHEN rating < 9.5 THEN 'Muy buena'
    ELSE 'Excelente'
  END AS rating_cat
FROM movies
ORDER BY rating
```

DISTINCT

16

```
SELECT DISTINCT actors.first_name, actors.last_name
FROM actors
    INNER JOIN actor_movie ON actors.id = actor_movie.actor_id
    INNER JOIN movies ON movies.id = actor_movie.movie_id
WHERE
    movies.title LIKE '%Harry Potter%';
```

nombre	apellido
▶ Daniel	Radcliffe
Emma	Watson
Helena	Bonham Carter
Rupert	Grint

Group by - Sintaxe

17

SELECT campo1 [,campo2,...]

FROM tabela

[where condições]

GROUP BY campo1[, campo2, ...]

<https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html>

Group by - Sintaxe

18

SELECT Marca

FROM Carros

GROUP by

Marca

id	Marca	Modelo	kilometros
1	Renault	Clio	10
2	Renault	Megane	23000
3	Seat	Ibiza	9000
4	Seat	Leon	20
5	Opel	Corsa	999
6	Renault	Clio	34000
7	Seat	Ibiza	2000
8	Seat	Cordoba	99999
9	Renault	Clio	88888

Marca
Opel
Renault
Seat

<https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html>

Having - sintaxe

19

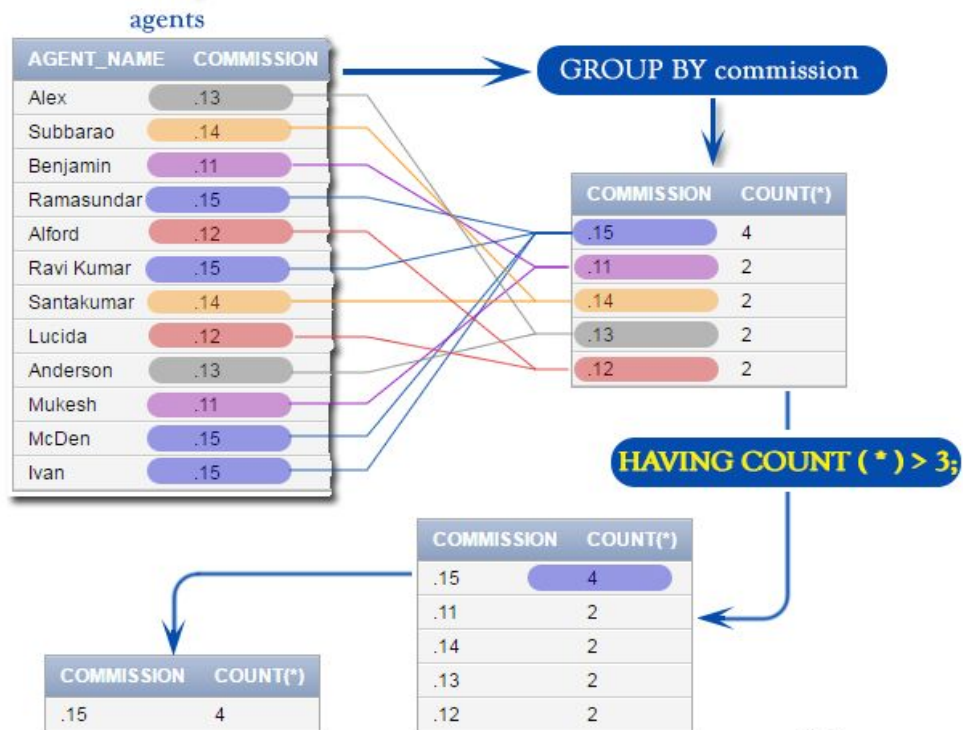
SELECT campo1 [,campo2, ...]

FROM tabela

GROUP BY campo1 [, campo2, ...]

HAVING condição

<https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html>



Having - sintaxe

21

SELECT commision,
count(*)

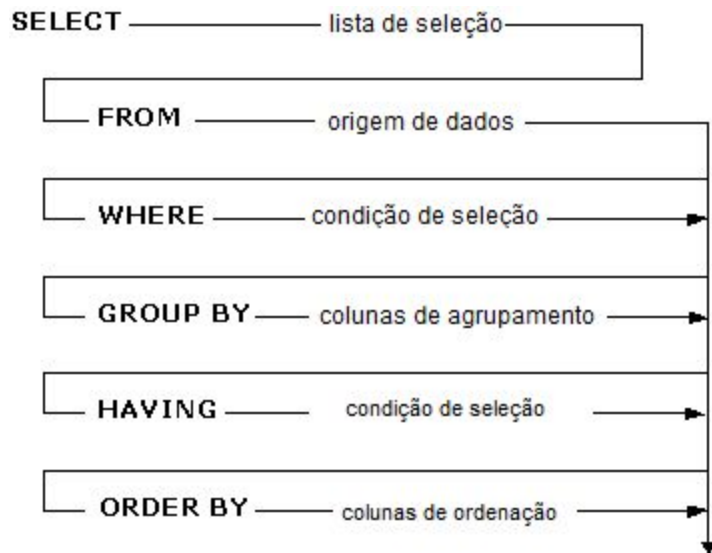
FROM **agents**

GROUP BY **commision**

HAVING **count(*)>3**

<https://dev.mysql.com/doc/refman/5.7/en/group-by-functions.html>

Estrutura ou ordem de uma Query ²²



Select **campos...**

From **tabela**

[Where condições]

[Group by colunas]

[Having condição]

[Order by colunas]

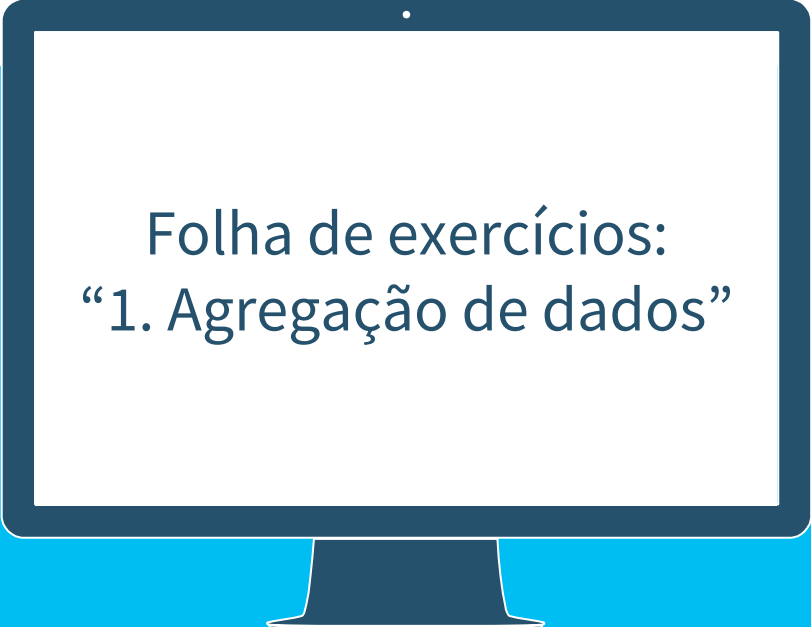
Vamos importar
um banco de
dados para
começar a
trabalhar com as
consultas.



Usando
MySQL
Workbench

Vamos executar
declarações

24



Folha de exercícios:
“1. Agregação de dados”

Subqueries

SUBQUERIES

26

Uma subquery é um SELECT dentro de outro SELECT. Também é conhecida como Inner Query ou Inner Select. Nas subqueries, podemos invocar uma tabela pela segunda vez.

SUBQUERIES - Sintaxe

27

```
SELECT campos...  
FROM tabela1  
WHERE campo1 in (  
    SELECT campoA  
    FROM tabelaA  
)
```

```
SELECT campos..., (  
    SELECT campoA  
    FROM tabelaA  
) [AS aliasCampo]  
FROM tabela1
```

SUBQUERIES - Sintaxe

28

```
SELECT campos...,  
[alias.campoA]  
FROM tabela1, (  
    SELECT campoA  
    FROM tabelaA  
) [alias]  
WHERE          condição
```

SUBQUERIES

29

Obter o(s) filme(s) com classificação inferior à média

```
SELECT id, title, rating
FROM movies
WHERE rating > (
    SELECT AVG(rating)
    FROM movies
);
```

SUBQUERIES

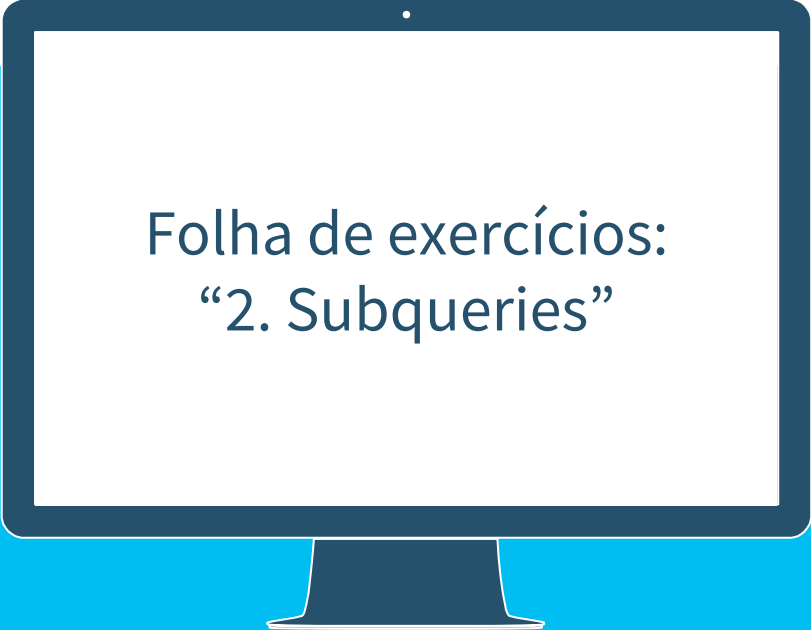
30

Obter os atores, o número de episódios e o número de filmes em que trabalharam.

```
SELECT
    a.id, a.first_name, a.last_name,
    (SELECT COUNT(*) FROM actor_movie AS am WHERE am.actor_id = a.id)
    AS tot_películas,
    (SELECT COUNT(*) FROM actor_episode AS ae WHERE ae.actor_id = a.id)
    AS tot_episodios
FROM actors AS a
```

Vamos executar
declarações

31



Folha de exercícios:
“2. Subqueries”

Exists

EXISTS

33

Operador de comparação.

Utilizado na cláusula Where.

Validar ou Negar uma condição.

EXISTS - Sintaxe

34

SELECT campos...

FROM tabela

WHERE

[NOT] EXISTS (subquery)

EXISTS - Sintaxe

35

```
SELECT campos...  
FROM tabela  
WHERE  
[NOT] EXISTS ( subquery )
```

```
SELECT coluna1  
FROM tabela1  
WHERE  
EXISTS  
(SELECT * FROM tabela2);
```

EXISTS - Sintaxe

36

Obter os atores cujos filmes preferidos durem 2h.

```
SELECT a.first_name, a.last_name
FROM actors AS a
WHERE EXISTS (
    SELECT m.id FROM movies AS m
    WHERE m.id = a.favorite_movie_id
    AND length = 120
);
```

Vamos executar
declarações

37



Folha de exercícios:
“3. Exists”