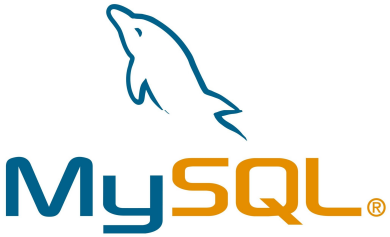
MySQL



Experticia en Querys

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

Agrupación de datos

Problemas a resolver:

- ¿Cuántas películas tengo?
- ¿Cuántas películas tengo del género X?
- ¿Cuál es el rating promedio de peliculas?
- ¿Cuál es el mínimo/máximo rating que tiene una película?

Funciones de agregación

- COUNT
- MIN
- MAX
- SUM
- AVG

COUNT


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

SELECT COUNT(id)
FROM movies;

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

AVG, SUM, MIN, MAX


```
SELECT AVG(rating)
FROM movies;

SELECT SUM(length)
FROM movies;

SELECT MIN(rating)
FROM movies;

SELECT MAX(rating)
FROM movies;
```

Más funciones interesantes

- 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


```
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

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


```
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


```
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


```
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


```sql
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


```
 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
```

Result Grid | Filter Rows: | 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


```sql
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


```
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 - Sintaxis

SELECT campo1 [,campo2,...]

FROM tabla

[where condiciones]

GROUP BY campo1[, campo2, ...]

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

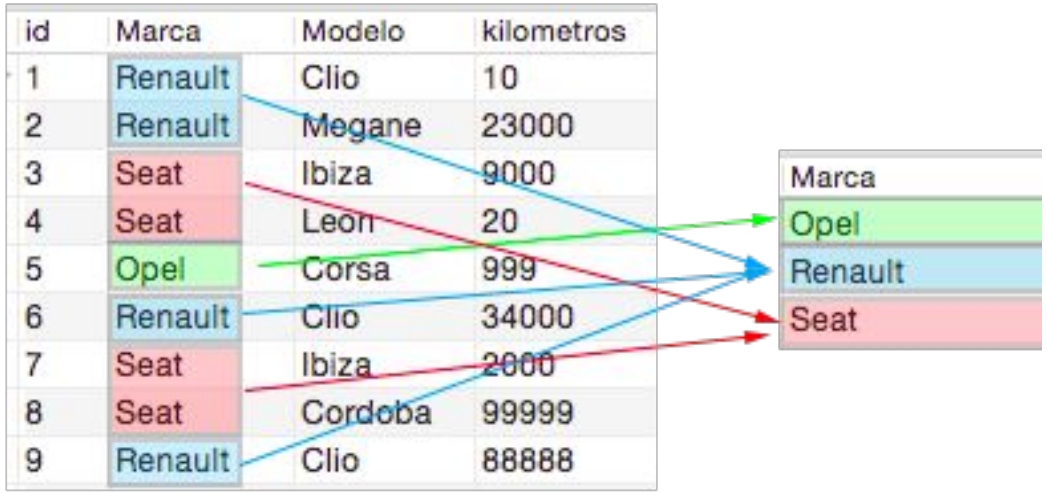
Group by - Sintaxis

SELECT Marca

FROM Autos

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 - sintaxis

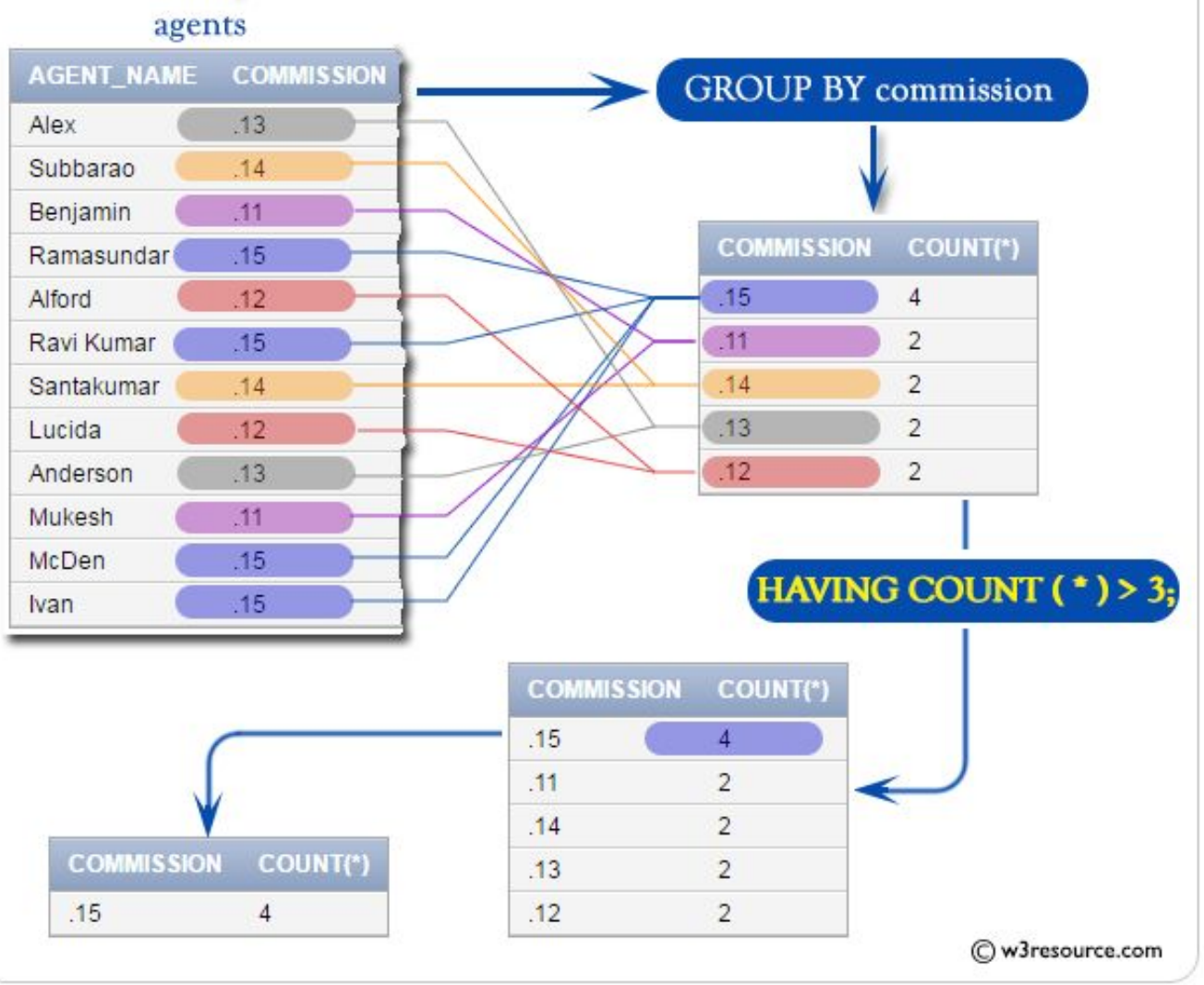
SELECT campo1 [,campo2, ...]

FROM tabla

GROUP BY campo1 [, campo2, ...]

HAVING condición

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


agents
AGENT_NAME COMMISSION
Alex .13
Subbarao .14
Benjamin .11
Ramasundar .15
Alford .12
Ravi Kumar .15
Santakumar .14
Lucida .12
Anderson .13
Mukesh .11
McDen .15
Ivan .15
GROUP BY commission
COMMISSION COUNT(*)
.15 4
.11 2
.14 2
.13 2
.12 2
HAVING COUNT (*) > 3;
COMMISSION COUNT(*)
.15 4
.11 2
.14 2
.13 2
.12 2
COMMISSION COUNT(*)
.15 4
© w3resource.com

Having - sintaxis

SELECT commision, count(*)

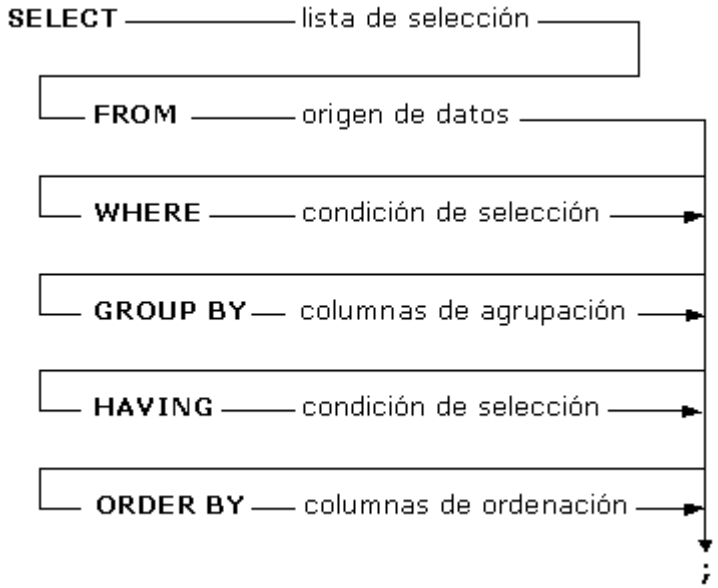
FROM **agents**

GROUP BY **commision**

HAVING **count(*)>3**

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

Estructura u orden de un Query



Select **campos**...

From **tabla**

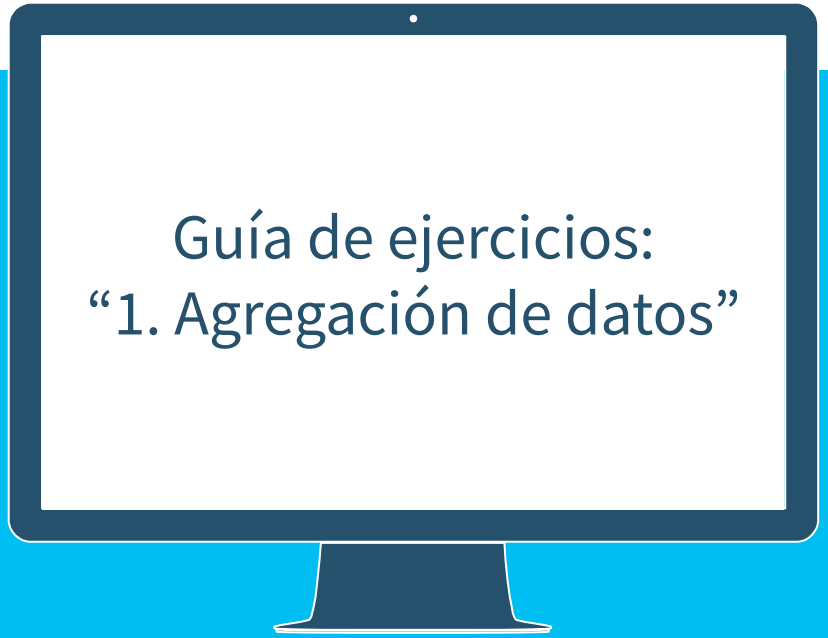
[Where condiciones**]**

[Group by columnas]

[Having condición]

[Order by columnas]

Ejecutemos sentencias



Subqueries

SUBQUERIES

Un subquery es un SELECT dentro de otro SELECT.

Tambien llamado Inner Query o Inner Select.

En los subqueries podemos llamar por segunda vez a la misma tabla.

SUBQUERIES - Sintaxis


```
SELECT  campos…
FROM tabla1
WHERE campo1 in (
  SELECT campoA
    FROM tablaA
)
```

```
SELECT  campos…, (
  SELECT campoA
    FROM tablaA
) [AS aliasCampo]
FROM tabla1
```

SUBQUERIES - Sintaxis


```
SELECT  campos…,
[alias.campoA]
FROM tabla1, (
  SELECT campoA
    FROM tablaA
) [alias]
WHERE              condicion
```

SUBQUERIES

Traer la(s) pelicula(s) que tengan un rating mejor que el promedio

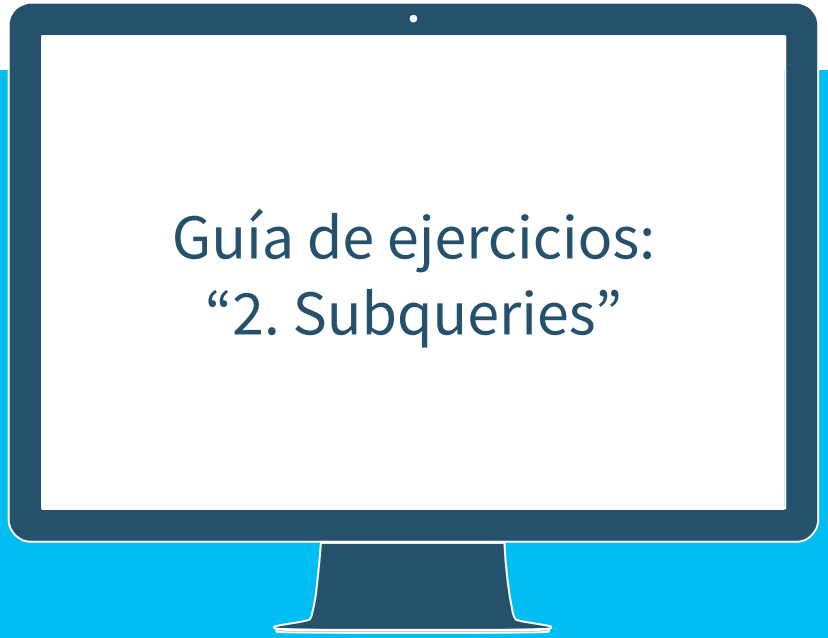
```
SELECT id, title, rating
FROM movies
WHERE rating > (
    SELECT AVG(rating)
    FROM movies
    );
```

SUBQUERIES

Obtener los actores, junto con la cantidad de episodios y la cantidad de películas en la que actuaron.

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

Ejecutemos sentencias



Exists

EXISTS

Operador de comparación.

Se utiliza en la cláusula Where.

Validar o Negar una condición.

EXISTS - Sintaxis

SELECT campos…

FROM tabla

WHERE

[NOT] EXISTS (subquery)

EXISTS - Sintaxis


```
SELECT campos…
FROM                    tabla
WHERE
[NOT] EXISTS ( subquery )
```

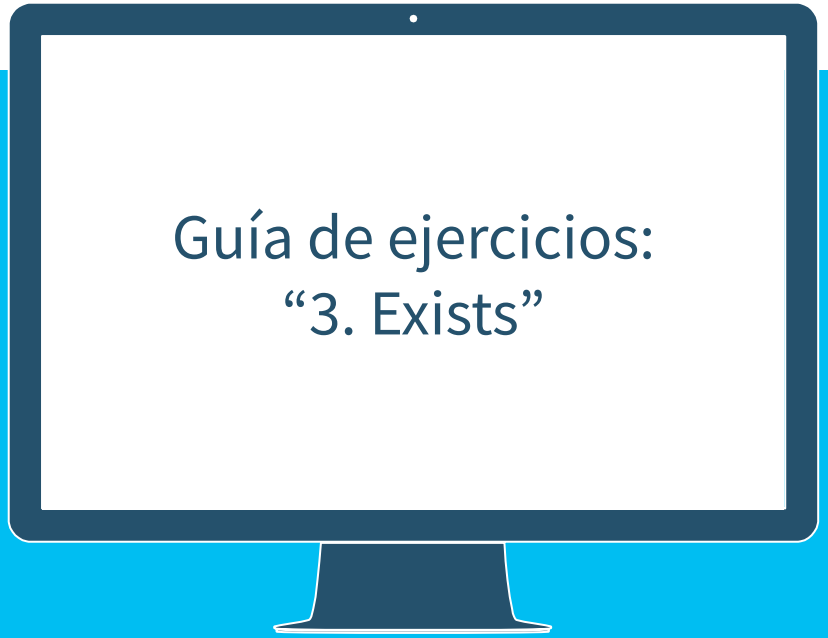
```
SELECT columna1
 FROM tabla1
 WHERE
 EXISTS
 (SELECT * FROM tabla2);
```

EXISTS - Sintaxis

Obtener los actores cuyas películas preferidas duren 2hs.

```
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
    );
```

Ejecutemos sentencias



GRACIAS

¿Que vimos hoy?

¿Preguntas?