

# Consultas SQL 2

IT BOARDING

**BOOTCAMP**



# Índice



**01** Repaso

**02** Join

**03** Group By

**04** Having

**05** Subconsultas

IT BOARDING

**BOOTCAMP**

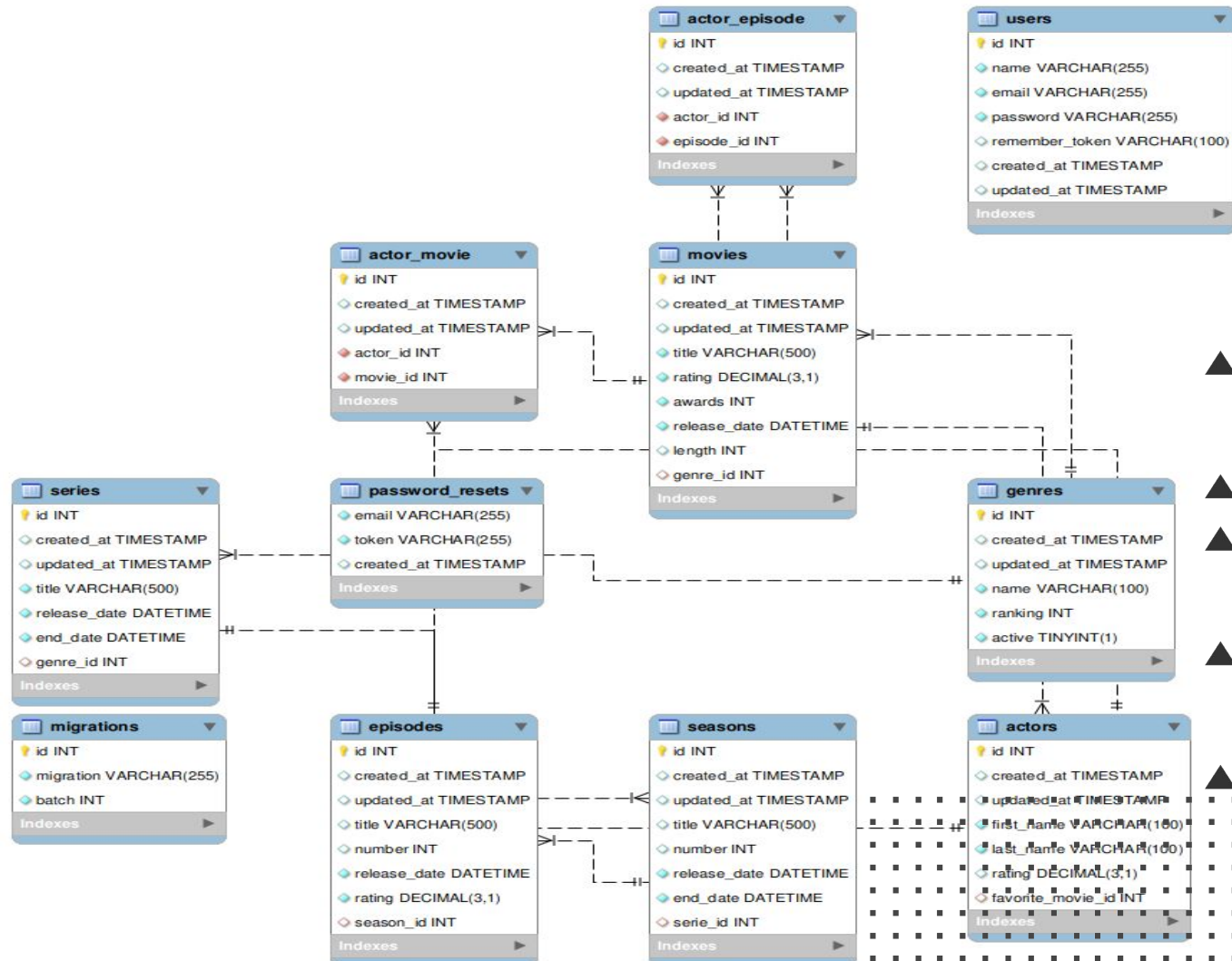
**En este módulo vamos a explicar conceptos teóricos y prácticos referidos a las consultas en SQL.**

IT BOARDING

**BOOTCAMP**

# Movies DB

Base de datos de ejemplo



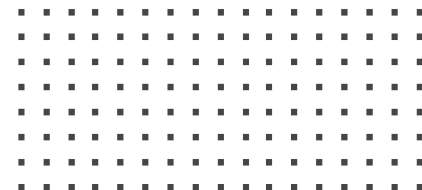
# REPASO

2021

funciones de agregación

```
SELECT count(*) as cantidad  
FROM actors  
WHERE rating = 7.5  
AND favorite_movie_id = 1;
```

filtros con where y and

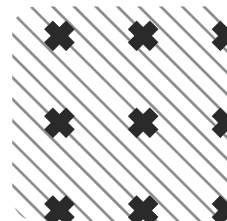
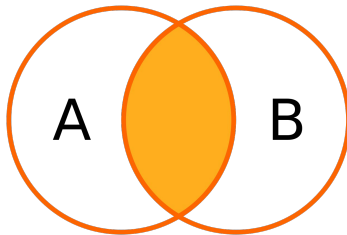


# JOIN (INNER JOIN)



La sentencia **INNER JOIN** es la sentencia **JOIN** por defecto, y consiste en combinar datos de una tabla con datos de la otra tabla a partir de una o varias **condiciones en común**.

INTERSECCIÓN ( $A \cap B$ )



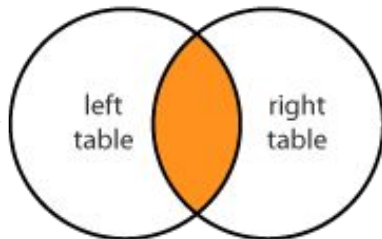
# TIPOS DE JOIN

2021

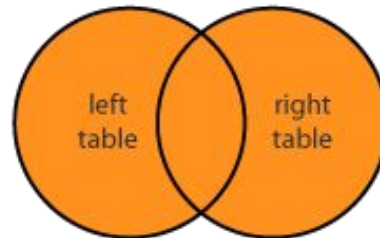
**INTERSECCIÓN ( $A \cap B$ )**

**El JOIN más utilizado**

INNER JOIN



FULL JOIN

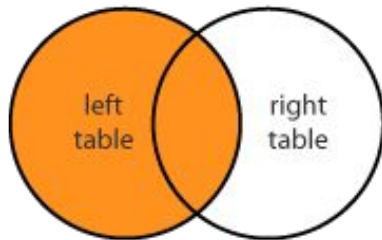


**UNIÓN ( $A \cup B$ )**

**LEFT JOIN +  
RIGHT JOIN**

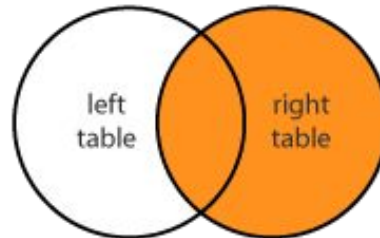
Muy poco utilizado. Se busca obtener la información de AMBAS tablas

LEFT JOIN



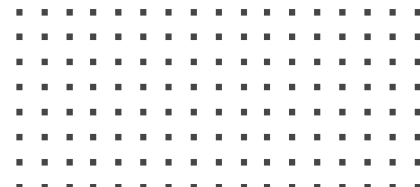
**DIFERENCIA ( $A - B$ )**

RIGHT JOIN



**DIFERENCIA ( $B - A$ )**

**Después del INNER JOIN son los más utilizados.**





## Sintaxis + Ejemplos



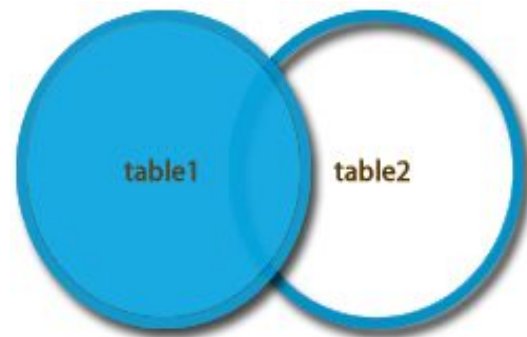
```
SELECT mo.*,  
       ac.first_name,  
       ac.last_name  
FROM movies mo  
INNER JOIN actors ac  
ON mo.id = ac.favorite_movie_id;
```





## Left Join

```
SELECT *  
FROM movies mo  
LEFT JOIN actors ac  
ON mo.id = ac.favorite_movie_id;
```





## Group by

- Agrupa los resultados según las columnas indicadas.
- Genera un solo registro por cada grupo de filas que compartan la columnas indicadas.
- Reduce la cantidad de filas de la consulta.
- Se suele utilizar en conjunto con **funciones** de **agregación**, para obtener **datos resumidos** y **agrupados** por las **columnas** que se necesiten.

# Group by

## ¿Cuánto gastó cada persona en total?

SKU (PK)	DNI	FECHA	PRECIO
1	33.241.677	01/01/2017	50
2	35.186.928	02/01/2017	60
3	33.241.677	03/01/2017	70
4	35.186.928	04/01/2017	40

## ¿Cómo imaginamos el reporte?

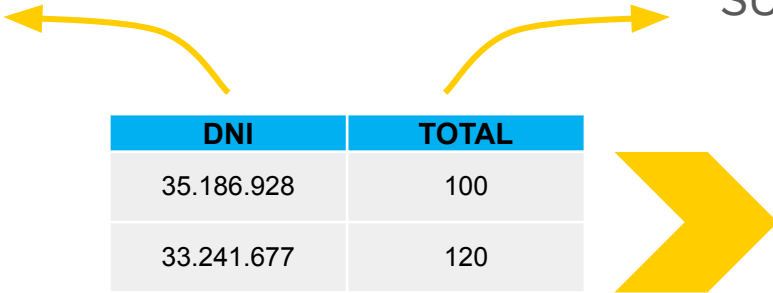


# Group by

¿Cómo podemos deducir la consulta?

**SELECT** DNI

SUM(PRECIO)



DNI	TOTAL
35.186.928	100
33.241.677	120

**GROUP BY DNI**



# Group by

## ¿Cómo funciona?

Agrupando por DNI, se crean grupos diferentes por cada DNI que exista en la tabla.

En este ejemplo, existen dos grupos:

**AGRUPACIÓN**

SKU (PK)	DNI	FECHA	PRECIO
1	33.241.677	01/01/2017	50
2	35.186.928	02/01/2017	60
3	33.241.677	03/01/2017	70
4	35.186.928	04/01/2017	40

SKU (PK)	DNI	FECHA	PRECIO
1	33.241.677	01/01/2017	50
3	33.241.677	03/01/2017	70

SKU (PK)	DNI	FECHA	PRECIO
2	35.186.928	02/01/2017	60
4	35.186.928	04/01/2017	40




# Group by

## ¿Cómo funciona?

Sobre cada grupo, se aplica la función de agregación que se indicó en el SELECT.

En este caso, se aplica la función SUMA sobre la columna precio.

## FUNCIÓN AGREGACIÓN



SKU (PK)	DNI	FECHA	PRECIO
2	35.186.928	02/01/2017	60
4	35.186.928	04/01/2017	40

SKU (PK)	DNI	FECHA	PRECIO
1	33.241.677	01/01/2017	50
3	33.241.677	03/01/2017	70

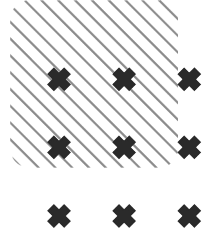
## Group by

### ¿Cómo funciona?

El resultado de la consulta, es una tabla que contiene el resultado de cada grupo.

#### 3 RESULTADO

DNI	TOTAL
35.186.928	100
33.241.677	120





## Group by: SINTAXIS + EJEMPLO



```
SELECT count(*),  
       mo.title,  
       mo.rating,  
       mo.awards  
FROM movies mo  
INNER JOIN actors ac ON mo.id = ac.favorite_movie_id  
GROUP BY title;
```





## {having}

Es muy **similar** a la cláusula **WHERE**, pero en lugar de **afectar a las filas de la tabla, afecta a los grupos obtenidos por el GROUP BY.**





# Having

Se desea obtener solo las persona que realizaron compras por un total superior a 100.

## ORIGINAL

DNI	TOTAL
35.186.928	100
33.241.677	120



## FINAL

DNI	TOTAL
35.186.928	100
33.241.677	120

Se desea obtener solo las persona que realizaron compras por un total superior a 100.

## ORIGINAL

DNI	TOTAL
35.186.928	100
33.241.677	120



## FINAL

DNI	TOTAL
33.241.677	120

**HAVING** TOTAL > 100

}

# Having


## Consulta completa

Se desea obtener solo las persona que realizaron compras por un total superior a 100.

**SELECT**

DNI

SUM(PRECIO)



DNI	TOTAL
35.186.928	100
33.241.677	120

**HAVING** TOTAL > 100



## Having sintaxis + ejemplo

```
SELECT count(*) as tot_act,  
       mo.title,  
       mo.rating,  
       mo.awards  
FROM movies mo  
INNER JOIN actors ac ON mo.id = ac.favorite_movie_id  
GROUP BY title HAVING tot_act > 2;
```




## Group by - having - where

Se desea obtener solo las persona que realizaron compras por un total superior a 100, pero considerando que cada compra individual haya sido superior a 50.

**SELECT**

DNI

SUM(PRECIO)



SKU (PK)	DNI	FECHA	PRECIO
1	33.241.677	01/01/2017	50
2	35.186.928	02/01/2017	60
3	33.241.677	03/01/2017	70
4	35.186.928	04/01/2017	40



## Group by - having - where

Se desea obtener solo las persona que realizaron compras por un total superior a 100, pero considerando que cada compra individual haya sido superior a 50.

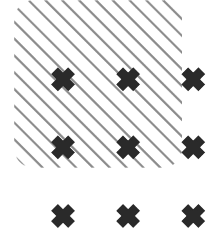
1

**WHERE** precio > 50**SELECT**

DNI

SUM(PRECIO)

SKU (PK)	DNI	FECHA	PRECIO
1	33.241.677	01/01/2017	50
2	35.186.928	02/01/2017	60
3	33.241.677	03/01/2017	70
4	35.186.928	04/01/2017	55



## Group by - having - where

Se desea obtener solo las persona que realizaron compras por un total superior a 100, pero considerando que cada compra individual haya sido superior a 50.

2

### AGRUPACIÓN

SKU (PK)	DNI	FECHA	PRECIO
1	33.241.677	01/01/2017	50
2	35.186.928	02/01/2017	60
3	33.241.677	03/01/2017	70
4	35.186.928	04/01/2017	55

SKU (PK)	DNI	FECHA	PRECIO
3	33.241.677	03/01/2017	70

SKU (PK)	DNI	FECHA	PRECIO
2	35.186.928	02/01/2017	60
4	35.186.928	04/01/2017	40



# Group by - having - where

Se desea obtener solo las persona que realizaron compras por un total superior a 100, pero considerando que cada compra individual haya sido superior a 50.

## FUNCIÓN AGREGACIÓN

SUM(PRECIO)

SKU (PK)	DNI	FECHA	PRECIO
2	35.186.928	02/01/2017	60
4	35.186.928	04/01/2017	55

SUM(PRECIO)

SKU (PK)	DNI	FECHA	PRECIO
3	33.241.677	03/01/2017	70



## Group by - having - where

Se desea obtener solo las persona que realizaron compras por un total superior a 100, pero considerando que cada compra individual haya sido superior a 50.

**HAVING TOTAL > 100**

DNI	TOTAL
35.186.928	115
33.241.677	70



## Group by - having - where

Se desea obtener solo las persona que realizaron compras por un total superior a 100, pero considerando que cada compra individual haya sido superior a 50.

4

### RESULTADO

DNI	TOTAL
35.186.928	115

## Group by - having - where

### ORDEN DE EJECUCIÓN





## Group by - having - where

### EJEMPLO + SINTAXIS



```
SELECT awards,  
       count(*)  
FROM movies WHERE rating > 8  
GROUP BY awards HAVING awards > 2  
order by awards DESC;
```



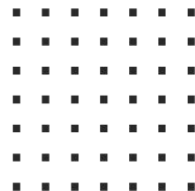
## Subconsultas

A la hora de resolver queries complejas se pueden utilizar subqueries, es decir, obtener resultados basados en resultados previos obtenidos a través de otra consulta.

# Subconsultas

## ¿QUÉ SON Y PARA QUÉ SIRVEN?

- Una subquery es una sentencia SELECT que aparece dentro de otra sentencia SELECT que llamaremos consulta principal.
- Restringir un conjunto limitado de datos en una nueva tabla.
- Filtrar datos en una consulta.



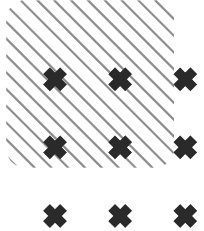


# Subconsultas

## SINTAXIS + EJEMPLO

```
SELECT *  
FROM actor_movie  
WHERE movie_id IN (SELECT id from movies  
WHERE rating=9.0);
```





## Buenas Prácticas

- Validar que el campo por el cual hago el **JOIN** sean del **mismo tipo de dato**.
- Evitar aplicar **funciones** sobre los campos por los cuales hago un **JOIN**.
- Revisar **alias** en los **JOINS** para **evitar productos cartesianos** en las consultas.
- Evitar uso excesivo de **JOINS**.
- Evitar el uso de **subqueries** en tablas de **gran volumen** que no utilicen algún **índice**.
- **Reescribir** la consulta validando los **JOINS**, **FILTROS**, a veces es mejor un **DISTINCT** en lugar de un **GROUP BY**.



# Gracias.

IT BOARDING

**BOOTCAMP**

