# MYSQL

## Creación de la base de datos

Ejemplo de creación de base de datos ejercicio 1:

**CREATE DATABASE** exercici1;

Ejemplo de creación de base de datos del ejercicio 1 comprobando si ya existía previamente:

**CREATE DATABASE IF NOT EXISTS** exercici1;

Apuntar a la base de datos con la que queremos trabajar:

**USE** exercici1;

Mostrar los juegos de caracteres:

**SHOW CHARACTER SET;**

Mostrar las colaciones disponibles para todos los juegos de caracteres

**SHOW COLLATION;**

Mostrar todas las colaciones disponibles por el charset latin1:

**SHOW COLLATION LIKE 'latin1%';**

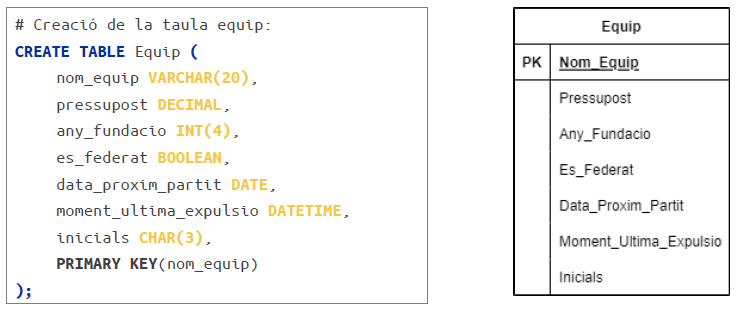
Crear la base de datos con juego de caracteres utf8 y compaginación binaria:

**CREATE DATABASE** exercici1**DEFAULT CHARACTER SET 'utf8' DEFAULT COLLATE 'utf8\_bin';**

Crear la base de datos con juego de caracteres utf8 y compaginación española no sensible a mayúsculas:

**CREATE DATABASE IF NOT EXISTS** ejercicio1**DEFAULT CHARACTER SET 'utf8' DEFAULT COLLATE 'utf8\_spanish\_ci';**

# Creación de tablas



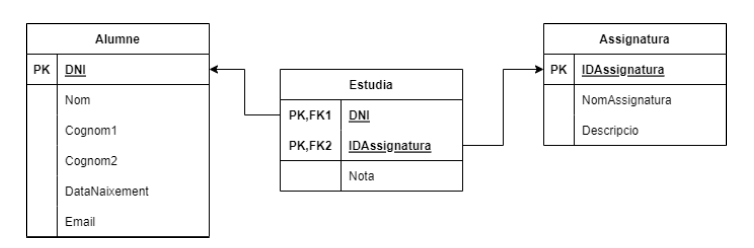
Para crear la tabla comprobando si existe escribir: **IF NOT EXISTS** entre table y el nombre

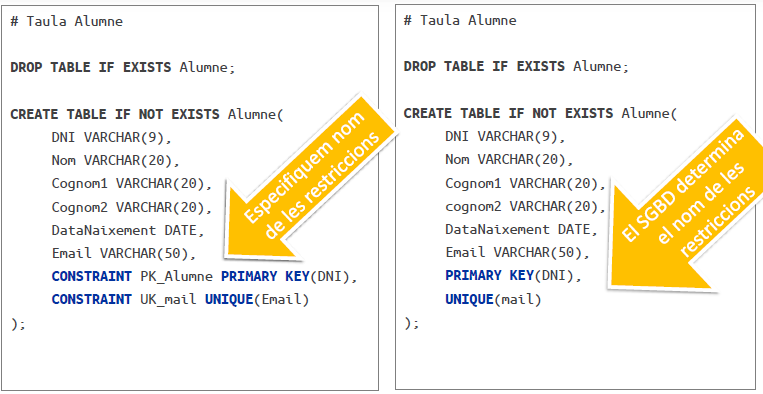
Eliminar una tabla

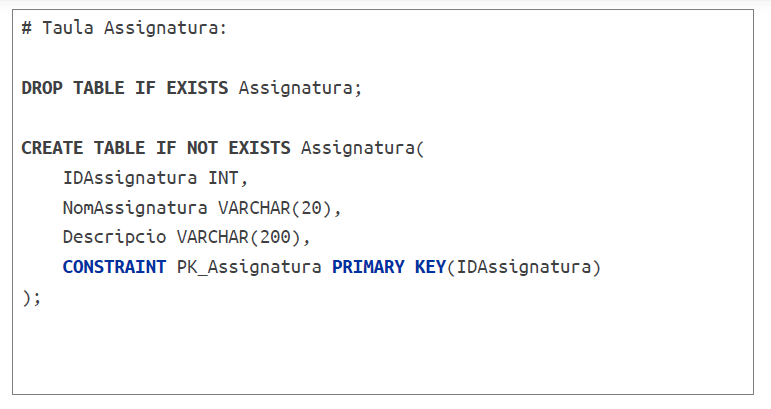
DROP TABLE Alumne;

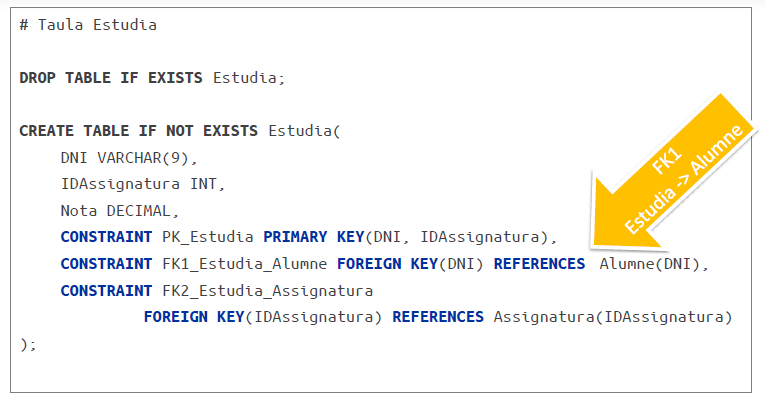
Eliminar una tabla si ya existe

DROP TABLE IF EXISTS Alumne;









# Propiedades de la creación de tablas

Auto-Increment

Solo se puede usar si se cumplen 4 condiciones:

* El atributo es un INT
* El atributo es PK
* La PK es simple
* El atributo no forma parte de una FK al mismo tiempo

# TaulaAssignatura:

DROP TABLE IF EXISTS Assignatura;

CREATE TABLE IF NOT EXISTS Assignatura(

* IDAssignaturaINT **AUTO\_INCREMENT**,
* NomVARCHAR(20),
* DescripcióVARCHAR(200),
* CONSTRAINT PK\_Asignatura PRIMARYKEY(IDAsignatura)

);

# Taula Assignatura:

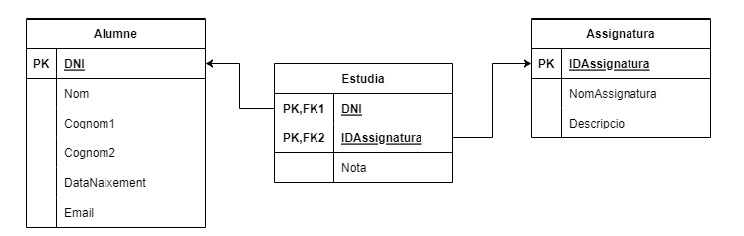
DROP TABLE IF EXISTS Assignatura;

CREATE TABLE IF NOT EXISTS Assignatura(

* IDAssignatura **INT UNSIGNED ZEROFILL AUTO\_INCREMENT NOT NULL**,
* NomVARCHAR(20) **NOT NULL CHARACTER SET 'utf8' COLLATE 'utf8\_bin'**,,
* DescripcióVARCHAR(200) **DEFALULT 'Falta descripció'**,
* CONSTRAINT PK\_AssignaturaPRIMARY KEY(IDAssignatura)

);

# Alteración de las tablas



Añadir una PK a la tabla Asignatura

**ALTER TABLE** Assignatura

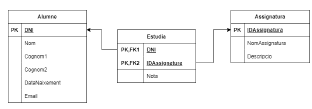
**ADD CONSTRAINT** PK\_Assignatura **PRIMARY KEY**(IDAssignatura);

Añadir una PK a la tabla Estudia

**ALTER TABLE** Estudia

**ADD CONSTRAINT** PK\_Estudia **PRIMARY KEY**(DNI, IDAssignatura);

Añadir una FK entre las tablas Estudia y Alumno

**ALTER TABLE** Estudia

**ADD CONSTRAINT** FK1\_Estudia\_Alumne

**FOREIGN KEY**(DNI) **REFERENCES** Alumne (DNI);

Añadir una FK entre las tablas Estudia y Asignatura

**ALTER TABLE** Estudia

**ADD CONSTRAINT** FK2\_Estudia\_Assignatura

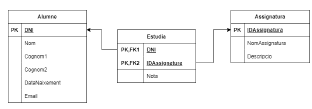
**FOREIGN KEY**(IDAssignatura) **REFERENCES**Assignatura(IDAssignatura);

Políticas de propagación de cambios

Cuando usamos UPDATE y DELETE hay que seguir unas normas de integridad.

* RESTRICT: No se permite modificar/eliminar un valor de la PK si este se encuentra referenciado en una FK.
* SET NULL: Se actualizan a NULL los valores de los FK que quedan huérfanos.
* CASCADE: Se actualizan o eliminan automáticamente los valores de los FK que se vean afectados por una modificación o eliminación de valores de una PK.

Definimos política de propagación de cambios

# Exemple definición de la FK entre les taules Estudia i Alumne:

**ALTER TABLE** Estudia

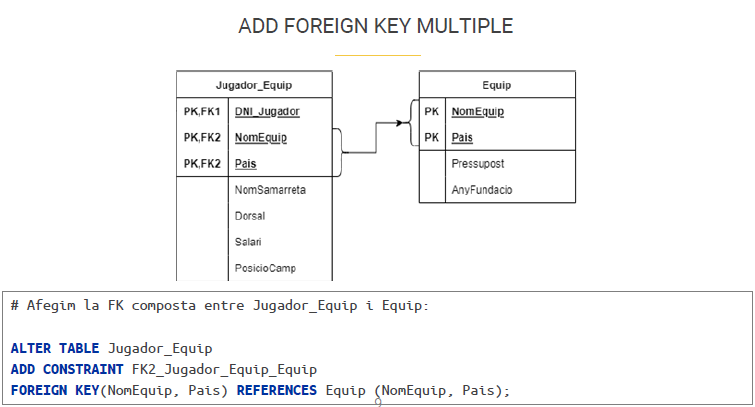
**ADD CONSTRAINT** FK1\_Estudia\_Alumne

**FOREIGN KEY**(DNI) **REFERENCES** Alumne (DNI)

**ON UPDATE CASCADEON DELETE RESTRICT**;

# Tot valor de DNI referenciat que es modifiqui a la taula Alumne, s’actualitzarà a la taula Estudia automàticament.

# No es permet eliminar de la taula Alumne un valor de DNI referenciat a la taula Estudia. No es pot eliminar un alumne que está estudiant.



ADD/DROP PK/FK

# Eliminar la PK de la tabla Assignatura:

**ALTER TABLE** Assignatura **DROP PRIMARY KEY**;

# Eliminar la FK entre las tablas Estudia i Assignatura:

**ALTER TABLE** Estudia **DROP FOREIGN KEY** FK2\_Estudia\_Assignatura;

ADD/DROP Columna

# Añadir columnas para guardar valores enteros positivos informando del número de créditos que tiene cada asignatura (por defecto 0) :

**ALTER TABLE** Assignatura **ADD** CreditsINT UNSIGNED DEFAULT 0;

# Eliminar la columna creada previamente:

**ALTER TABLE** Assignatura **DROP** Credits;

MODIFY

# Cambiar el tipo de datos i valor por defecto de la columna Descripcio:

**ALTER TABLE** Assignatura **MODIFY** Descripcio TEXT DEFAULT NULL;

# Cambiamos el tipaje de la columna NomAssignatura y restringimos valores NULL:

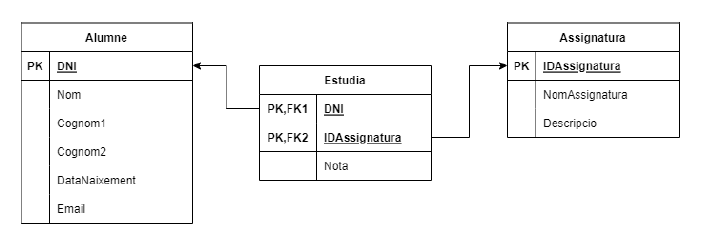
**ALTER TABLE** Assignatura **MODIFY** NomAssignatura VARCHAR(30) NOT NULL;

CHANGE

# Cambiar el nombre de la columna DNI de la tabla Estudia por el nombre de DNI\_Alumne(Hay que introducir el tipo de dato nuevo):

**ALTER TABLE** Estudia **CHANGE** DNI DNI\_Alumne INT(9) NOT NULL;

# RENAME Y DROP



RENAME

Cambiamos el nombre de la tabla Estudia (dos formas de hacerlo):

**RENAME TABLE** Estudia **TO** Alumne\_Assignatura;

**ALTER TABLE** Estudia **RENAME AS** Alumne\_Assignatura;

DROP

Elimina la columna Nota de la taula Estudia (es perden dades i estructura):

**ALTER TABLE** Estudia **DROP COLUMN** Nota;

Elimina la taula Estudia (es perden dades i estructura):

**DROP TABLE** Estudia;

# Elimina la taula Estudia si existeix:

**DROP TABLE IF EXISTS** Estudia;

DROP DATABASE

# Elimina la base de dades Exercici1:

**DROP DATABASE** Exercici1;

# Elimina la base de dades Exercici1 si existeix:

**DROP DATABASE IF EXISTS** Exercici1;

# VISTAS

VIEW

# Crea una vista amb els Noms i Cognoms dels Alumnes:

**CREATE VIEW** VistaAlumnes **AS** SELECT Nom, Cognom1 FROM Alumne;

# Crea o sobreescriu la vista amb els Noms i Cognoms dels Alumnes:

**CREATE OR REPLACE VIEW** Vista Alumnes

**AS** SELECT Nom, Cognom1, Cognom2 FROM Alumne;

# Mostra tot el contingut de la VistaALumnes:

**SELECT \*** **FROM** VistaAlumnes;

# Mostra nomésla columna dels noms dels alumnes:

**SELECT** Nom **FROM** VistaAlumnes;

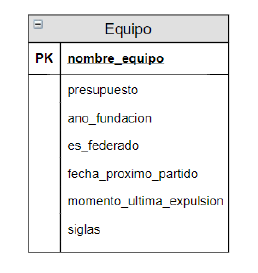
# Elimina la VistaAlumnes:

**DROP VIEW** VistaAlumnes;

# Elimina la VistaAlumnessi existeix:

**DROP VIEW IF EXISTS** VistaAlumnes;

## INSERCIÓN DE DATOS



Inserción de un equipo:

**INSERT INTO** equipo **VALUES**(‘Barcelona’, 3000000, 1899, TRUE, ‘2019-03-02’, ‘2018-11-10 21:03:27’, ‘FCB’);

**INSERT INTO** equipo **VALUES**(‘Sporting La Salle’, 5000, 2015, FALSE, ‘2019-03-20’, ‘2018-10-8 15:30:48’, ‘SLS’);

Insertar datos específicos equipos específicos

**INSERT INTO** equipo (nombre\_equipo, presupuesto, siglas) **VALUES** (‘Madrid’, 4000000, ‘RMA’);

Casos especiales



# Inserción de un equipo, jugador y relación entre ellos:

**INSERT INTO** equipo **VALUES**(‘Sporting La Salle’, 5000, 2000);

**INSERT INTO** jugador (DNI, nombre, apellido, nombre\_equipo) **VALUES**(‘22222222B’, ‘Juan’, ‘García’, ‘Sporting La Salle’);

**INSERT INTO** jugador (DNI, nombre, apellido, nombre\_equipo) **VALUES**(‘33333333C’, ‘David’, ‘López’, NULL);

**INSERT INTO** jugador (DNI, nombre, apellido, nombre\_equipo) **VALUES**(‘44444444D’, ‘Manuel’, ‘Expósito’, ‘Sevilla’); -> **ERROR!**



# Inserción de un alumno y relación entre ellos:

**INSERT INTO** alumno **VALUES**(‘11111111A’, ‘Paco’, ‘Pérez’, ‘Gutiérrez’, TRUE, ‘2000-07-22’);

**INSERT INTO** asignatura **VALUES** (1, ‘Bases de Datos’, NULL);

**INSERT INTO** estudia **VALUES** (‘11111111A’, 1, 7);

**INSERT INTO** estudia **VALUES**(‘11111111A’, 2, 7); -> **ERROR!**

# Inserciones con campo id autoincremental:

**INSERT INTO** asignatura **VALUES**(2, ‘Sistemas’, NULL);

**INSERT INTO** asignatura **VALUES**(NULL, ‘Sistemas’, NULL);

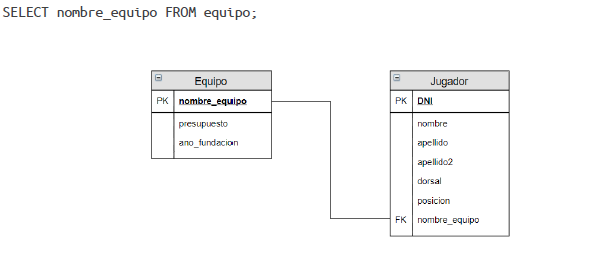
**INSERT INTO** asignatura **VALUES**(‘’, ‘JAVA’, NULL);

**INSERT INTO** asignatura (nombre, descripcion) **VALUES**(‘XML’, NULL);

# Multi-inserción de 2 asignaturas:

**INSERT INTO** asignatura **VALUES**(5, ‘FOL’, NULL)**,** (6, ‘EIE’, NULL);

# SELECTS BÁSICOS



OPERADORES DE COMPARACIÓN

# Menor que:

SELECT nombre\_equipoFROM equipo WHERE presupuesto **<**1000000;

# Mayor que:

SELECT nombre\_equipoFROM equipo WHERE presupuesto **>**1000000;

# Mayor o igual que:

SELECT nombre\_equipoFROM equipo WHERE presupuesto **>=**1000000;

# Menor o igual que:

SELECT nombre\_equipoFROM equipo WHERE presupuesto **<=**1000000;

# Igual que:

SELECT nombre\_equipoFROM equipo WHERE presupuesto **=**1000000;

# Igual que (texto):

SELECT DNI FROM jugador WHERE nombre **=**‘Juan’;

# Diferente que:

SELECT nombre\_equipoFROM equipo WHERE presupuesto **<>**1000000;

OPERACIONES DE COMPARACIÓN: LIKE

# Empieza por:

SELECT nombre\_equipo FROM equipo WHERE nombre\_equipo **LIKE** ‘B**%**’;

# No empieza por:

SELECT nombre\_equipo FROM equipo WHERE nombre\_equipo **NOT LIKE** ‘B**%**’;

# Termina por:

SELECT nombre\_equipo FROM equipo WHERE nombre\_equipo **LIKE** ‘**%**A’;

# Contiene:

SELECT nombre\_equipo FROM equipo WHERE nombre\_equipo **LIKE** ‘**%**A**%**’;

# Tiene 5 letras:

SELECT nombre\_equipo FROM equipo WHERE nombre\_equipo

**LIKE** ‘\_ \_ \_ \_ \_’;

# Tiene 5 letras donde la segunda es una A :

SELECT nombre\_equipo FROM equipo WHERE nombre\_equipo

**LIKE** ‘\_ A \_ \_ \_’;

# Tiene al menos dos letras (o mas):

SELECT nombre\_equipo FROM equipo WHERE nombre\_equipo

**LIKE** ‘\_ \_ %’;

# Igual que:

SELECT nombre\_equipo FROM equipo WHERE presupuesto **BETWEEN** 1000000 **AND** 2000000;

# Que cumple más de una condición:

SELECT nombre\_equipo FROM equipo WHERE presupuesto > 1000000 **AND** nombre\_equipo LIKE ‘B%’;

# Que cumple una u otra condición:

SELECT nombre\_equipo FROM equipo WHERE presupuesto > 1000000 **OR** nombre\_equipo LIKE ‘B%’;

# Que no cumple una condición:

SELECT nombre\_equipo FROM equipo WHERE **NOT**(presupuesto > 1000000)

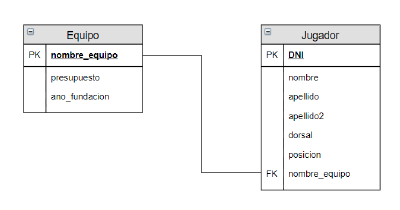
# Concatenación de operadores lógicos:

SELECT nombre\_equipo FROM equipo WHERE presupuesto > 1000000 **AND** nombre\_equipo LIKE ‘B%’ **AND** ano\_fundacion< 1900;

# Combinación de operadores lógicos:

SELECT nombre\_equipo FROM equipo WHERE (presupuesto > 1000000 **AND** ano\_fundación< 1900) **OR** (presupuesto < 1000000 **AND** ano\_fundacion> 1900);

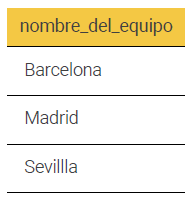
# Elementos auxiliares



**Alias**

# Alias en el select:

SELECT nombre\_equipo **AS** nombre\_del\_equipo FROM equipo;



# Alias en el from:

SELECT **e.**nombre\_equipo FROM equipo **AS** e WHERE **e.**presupuesto> 1000000;

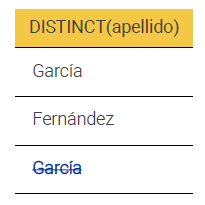
# Referencias sin alias:

SELECT **equipo.**nombre\_equipo FROM equipo WHERE **equipo.**presupuesto > 1000000;

**DISTINCT**

# Seleccionar apellidos distintos:

SELECT **DISTINCT**(apellido) FROM jugador;

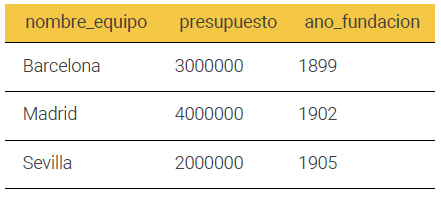


# Seleccionar apellidos distintos, combinado con un ALIAS

SELECT **DISTINCT**(apellido) **AS** apellidos FROM jugador;

**Operador \***

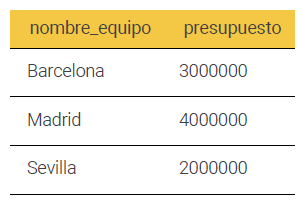
SELECT \* FROM equipo;

****

**Selecciones Multiatributo**

# Selección de mas de un atributo concreto (nombre y presupuesto):

SELECT nombre\_equipo, presupuesto FROM equipo;

****

**Selecciones con valores NULL**

# Selección de jugadores sin equipo:

SELECT \* FROM jugador WHERE nombre\_equipo **IS NULL**;

SELECT \* FROM jugador WHERE nombre\_equipo **= NULL**;

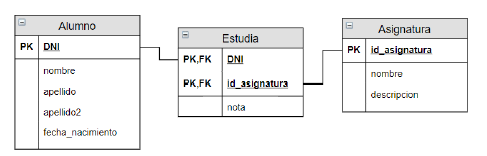
# Selección de jugadores con equipo:

SELECT \* FROM jugador WHERE nombre\_equipo **IS NOT NULL**;

SELECT \* FROM jugador WHERE nombre\_equipo **<> NULL**;

# FUNCIONES

**Funciones de fecha**

****

NOW

# Formato DATETIME:

**‘YYYY-MM-DD hh:mm:ss’**

# Seleccionar nombre y apellido de los alumnos con fecha de nacimiento menor a la fecha actual:

SELECT nombre, apellido FROM alumno WHERE fecha\_nacimiento < **NOW()**;

CURRENT\_DATE

# Formato DATE:

**‘YYYY-MM-DD’**

# Seleccionar nombre y apellido de los alumnos con fecha de nacimiento menor a la fecha actual:

SELECT nombre, apellido FROM alumno WHERE fecha\_nacimiento < **CURRENT\_DATE()**;

YEAR

# Mostrar en DNI y año de nacimiento de los alumnos:

SELECT DNI, **YEAR**(fecha\_nacimiento) FROM alumno;

# Mostrar el DNI de los alumnos nacidos antes del 1990:

SELECT DNI FROM alumno WHERE **YEAR**(fecha\_nacimiento) < 1990;

MONTH/DAY

# Mostrar el nombre y apellido de los alumnos nacidos en Enero:

SELECT nombre, apellido FROM alumno WHERE **MONTH**(fecha\_nacimiento) = 1;

# Mostrar el DNI de los alumnos que cumplen años en el mes actual:

SELECT DNI FROM alumno WHERE **MONTH**(fecha\_nacimiento) = **MONTH**(NOW());

SELECT DNI FROM alumno WHERE **MONTH**(fecha\_nacimiento) = **MONTH**(CURRENT\_DATE());

# Mostrar el nombre y apellido de los alumnos nacidos el primer día del mes:

SELECT nombre, apellido FROM alumno WHERE **DAY**(fecha\_nacimiento) = 1;

**FUNCIONES DE TEXTO**

**LENGTH**

# Nombre de asignatura tiene 5 letras:

SELECT nombre FROM asignatura WHERE nombre LIKE ‘\_ \_ \_ \_ \_’;

SELECT nombre FROM asignatura WHERE **LENGTH**(nombre) = 5;

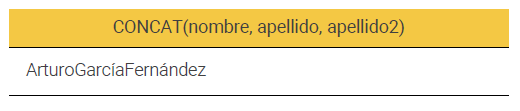
# Nombre de asignatura tiene mas de 5 letras:

SELECT nombre FROM asignatura WHERE **LENGTH**(nombre) > 5;

**CONCAT**

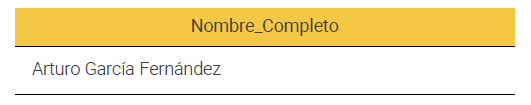
# Mostrar juntos en una columna el nombre, apellido y apellido2 del alumno con DNI 11111111A:

SELECT **CONCAT**(nombre, apellido, apellido2) FROM alumno WHERE DNI = ‘11111111A’



# Mostrar juntos en una columna el nombre, apellido y apellido2 del alumno con DNI 11111111A:

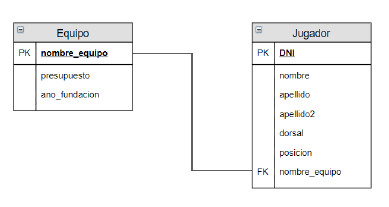
SELECT **CONCAT**(nombre, **‘ ’**, apellido, **‘ ’**, apellido2) AS **Nombre\_Completo** FROM alumno WHERE DNI = ‘11111111A’

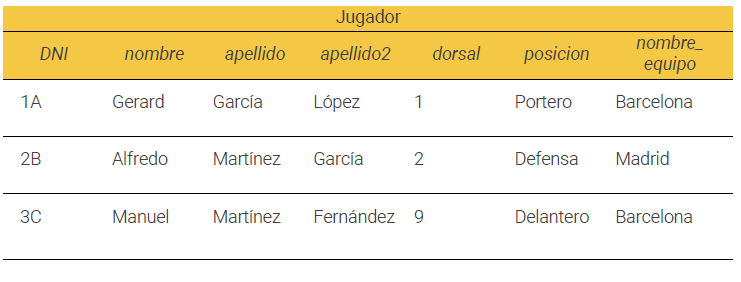


# ORDER BY

ASC (Por defecto)

DESC





# Selecciona el nombre y dorsal de los jugadores y ordena según su dorsal de forma ascendente:

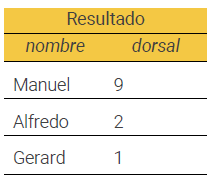
SELECT nombre, dorsal FROM jugador **ORDER BY** dorsal;

SELECT nombre, dorsal FROM jugador **ORDER BY** dorsal **ASC**;



# Selecciona el nombre y dorsal de los jugadores y ordena según su dorsal de forma descendente:

SELECT nombre, dorsal FROM jugador **ORDER BY** dorsal **DESC**;



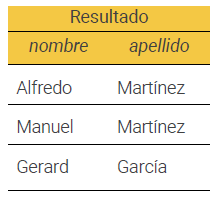
# Selecciona el nombre y ordena según los jugadores según su nombre de la A a la Z (ascendente):

SELECT nombre FROM jugador **ORDER BY** nombre **ASC**;



# Selecciona el nombre y apellido ordena los jugadores según su apellido de la Z a la A (descendente) y nombre de la A ala Z (ascendente):

SELECT nombre, apellido FROM jugador **ORDER BY** apellido**DESC ,** nombre **ASC**;



LIMIT

# Selecciona nombre y apellido de un jugador:

SELECT nombre, apellido FROM jugador **LIMIT** 1;

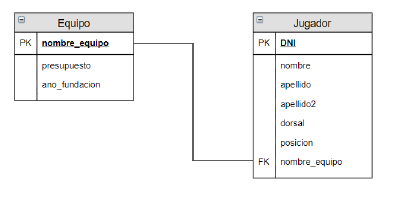
# Selecciona nombre y apellido de del primer jugador ordenando según el dorsal:

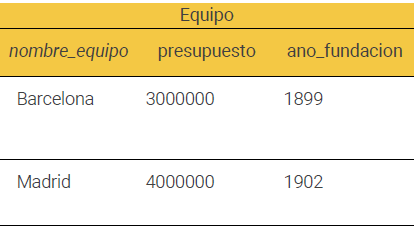
SELECT nombre, apellido FROM jugador ORDER BY dorsal ASC **LIMIT** 1;

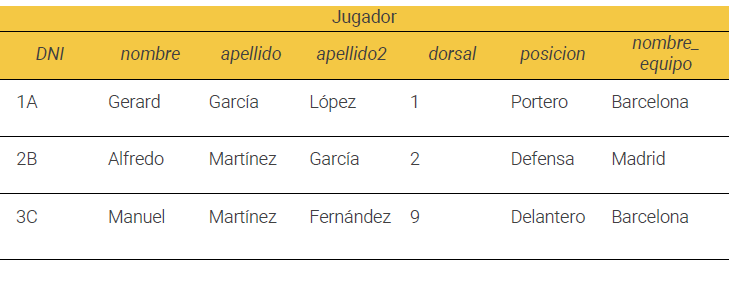
# Selecciona nombre y apellido de del tercer jugador ordenando según el dorsal:

SELECT nombre, apellido FROM jugador ORDER BY dorsal ASC **LIMIT** 2,1;

# GROUP BY Y FUNCIONES DE AGREGACIÓN

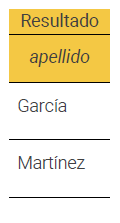






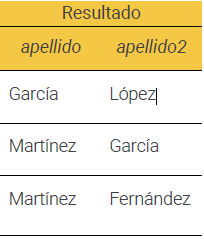
# Selección del apellido sin repeticiones:

SELECT apellido FROM jugador **GROUP BY** apellido;



# Selección de los apellidos sin repeticiones:

SELECT apellido, apellido2 FROM jugador **GROUP BY** apellido, apellido2;



**FUNCIONES DE AGREGACIÓN**

COUNT

# Cuantos jugadores hay por equipo:

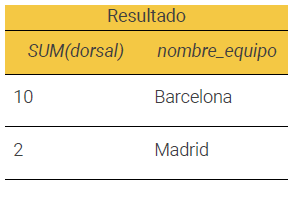
SELECT **COUNT**(DNI), nombre\_equipoFROM jugador **GROUP BY** nombre\_equipo;



SUM

# Suma valor de los dorsales por equipo:

SELECT **SUM**(dorsal), nombre\_equipoFROM equipo **GROUP BY** nombre\_equipo;



MIN

# Dorsal mas bajo del equipo:

SELECT **MIN**(dorsal), nombre\_equipoFROM equipo **GROUP BY** nombre\_equipo;



MAX

# Dorsal mas alto del equipo:

SELECT **MAX**(dorsal), nombre\_equipoFROM equipo **GROUP BY** nombre\_equipo;

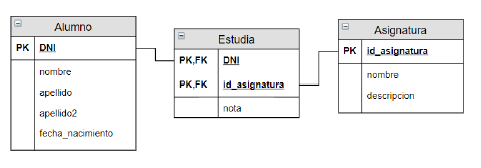


AVG

# Media de los presupuestos:

SELECT **AVG**(presupuesto) FROM jugador;

# HAVING



# Media de las notas de las asignaturas siempre que esta sea superior a 5:

SELECT AVG(nota), as.nombre FROM estudia AS e, asignatura AS as

WHERE e.id\_asignatura= as.id\_asignatura

GROUP BY as.id\_asignatura

**HAVING**AVG(nota) > 5

# Media de las notas de las asignaturas con una id entre 1 y 5:

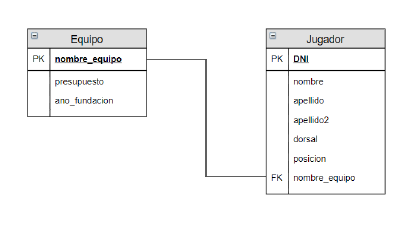
SELECT AVG(nota), as.nombre FROM estudia AS e, asignatura AS as

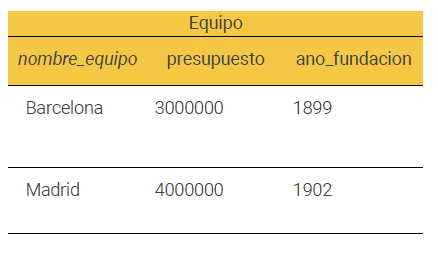
WHERE e.id\_asignatura= as.id\_asignatura

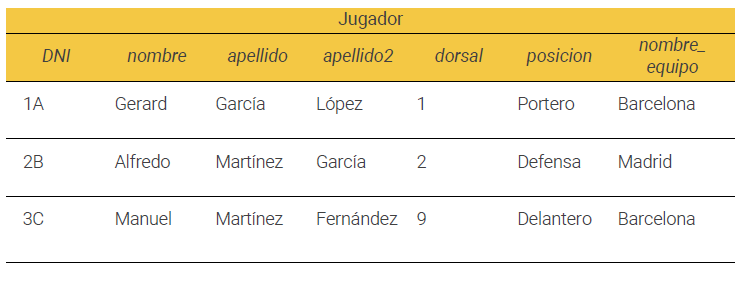
GROUP BY as.id\_asignatura

**HAVING** as.id\_asignatura BETWEEN 1 AND 5

# SELECCIONES MULTITABLA

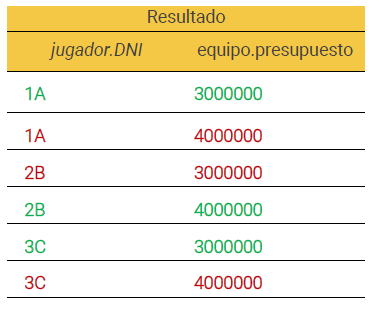






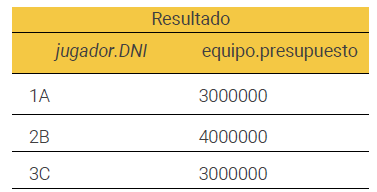
# Seleccionar DNI de los jugadores y el año de fundación de sus respectivos equipos:

SELECT DNI, ano\_fundacion FROM equipo, jugador;



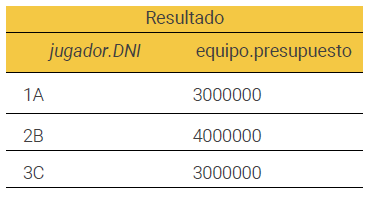
# Seleccionar DNI de los jugadores y el año de fundación de sus respectivos equipos:

SELECT DNI, ano\_fundacion FROM equipo, jugador WHERE **equipo.nombre\_equipo = jugador. nombre\_equipo**;



# Seleccionar DNI de los jugadores y el año de fundación de sus respectivos equipos:

SELECT DNI, ano\_fundacion FROM equipo AS e, jugador AS j WHERE **e.nombre\_equipo= j.nombre\_equipo**;



# Seleccionar los jugadores de los equipos fundados antes del 1900:

SELECT **j.\*** FROM equipo AS e, jugador AS j WHERE **e.nombre\_equipo= j.nombre\_equipo** ano\_fundacion<= 1900;

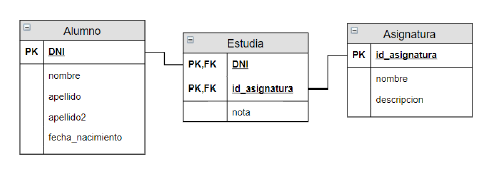


# Seleccionar los jugadores del Barcelona:

SELECT **j.\*** FROM jugador AS j WHERE **j.nombre\_equipo= ‘Barcelona’**;



SELECCIONES CON 3 TABLAS



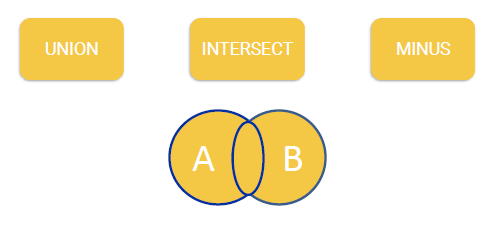
# Selecciona el nombre y apellido de los alumnos de la asignatura de Bases de Datos:

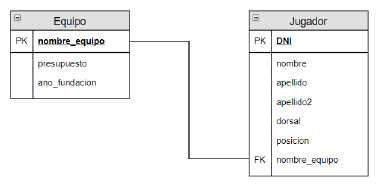
SELECT **a.**nombre, **a.**apellido FROM alumno AS a, estudia AS e, asignatura AS as WHERE

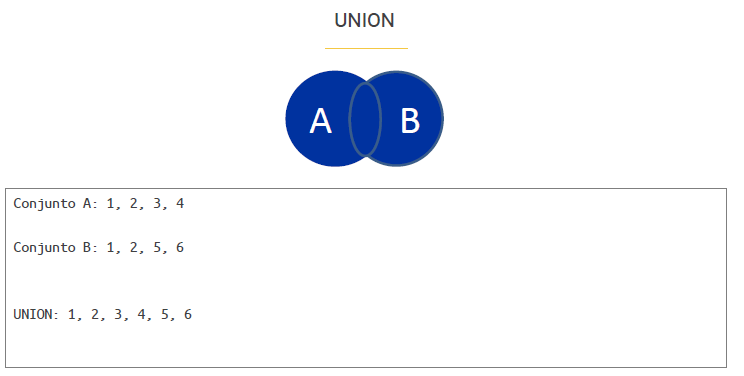
**a.DNI= e.DNI AND e.id\_asignatura= as.id\_asignatura**

AND **as.**nombre= ‘Bases de Datos’;

# Conjuntos





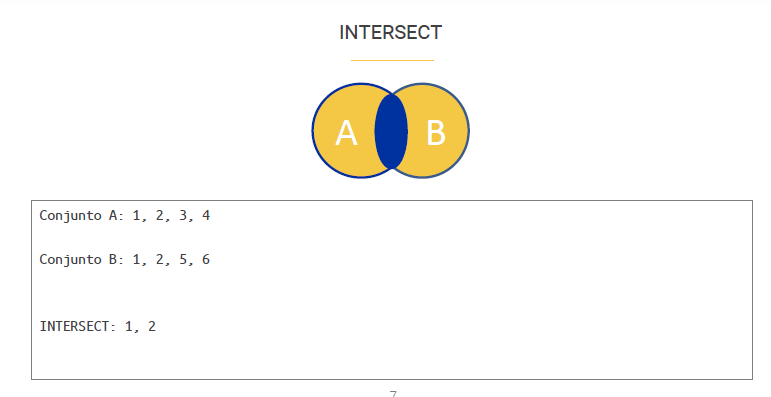


# Selecciona el nombre de los equipos fundados antes del año 1900 o después del 2000 (con conjuntos):

(SELECT nombre\_equipoFROM equipo WHERE ano\_fundacion< 1900)

**UNION**

(SELECT nombre\_equipoFROM equipo WHERE ano\_fundacion> 2000)

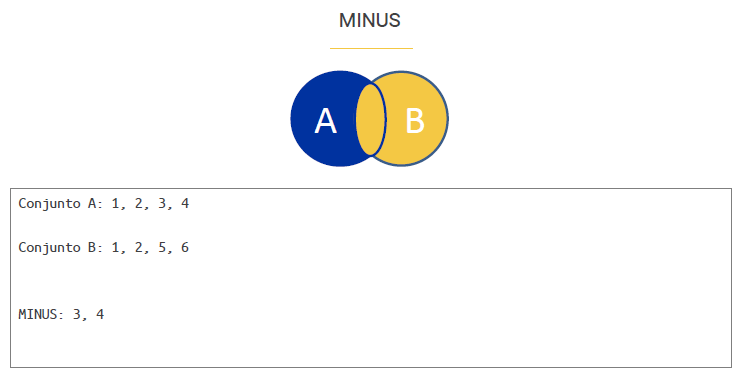


# Selecciona el nombre de los equipos fundados antes del año 1900 y con presupuesto mayor a 1000000 (con conjuntos):

(SELECT nombre\_equipoFROM equipo WHERE ano\_fundacion< 1900)

**INTERSECT**

(SELECT nombre\_equipoFROM equipo WHERE presupuesto > 1000000)



# Selecciona el nombre de los equipos fundados antes del año 1900 y que no tengan un presupuesto mayor a 3500000 (con conjuntos):

(SELECT nombre\_equipoFROM equipo WHERE ano\_fundacion< 1900)

**MINUS**

(SELECT nombre\_equipoFROM equipo WHERE presupuesto > 3500000)