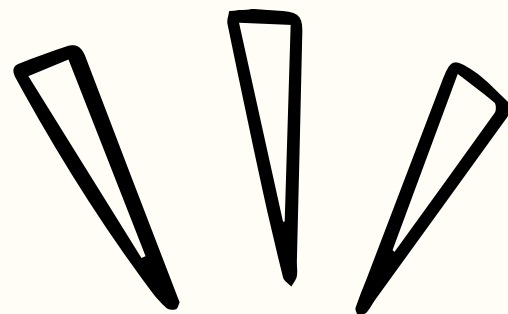


COMANDOS MYSQL





Lo primero que debemos hacer es crear
la base de datos

CREATE DATABASE HOSPITAL;

Luego debemos indicar que queremos
usar esa base de datos

USE HOSPITAL;

Ahora crearemos las tablas

Consejo: Empezar por las que no tienen foreign key

```
CREATE TABLE CONSULTORIO (ID_CONS INT PRIMARY KEY,  
NOMBRE VARCHAR(30),  
DIRECCION VARCHAR(50));
```

*Indico que este atributo
será la primary key*

*Indico el tipo de dato
y su longitud*

Si debo crear una tabla que su **foreign key** apunta a la creada anteriormente lo hago de la siguiente manera:

```
CREATE TABLE MEDICO (ID_MEDICO INT PRIMARY KEY,  
NOMBRE VARCHAR(30),  
ESPECIALIDAD VARCHAR(50),  
ID_CONSULTORIO_FK INT,  
FOREIGN KEY(ID_CONSULTORIO_FK) REFERENCES CONSULTORIO(ID_CONS));
```

Indico que es FK

*Señalo el atributo
que es FK*

*Indico con qué
tabla se relaciona*

*a qué atributo
de esa tabla*

! EL TIPO DE DATO DE ID_CONSULTORIO_FK Y EL DE ID_CONS DEBEN SER IGUALES

Cree una tabla y me olvide de indicar cuál es su Primary Key ¿Qué hago?

```
CREATE TABLE PACIENTE (DNI_PACIENTE BIGINT,  
NOMBRE VARCHAR(30),  
EDAD INT,  
DOMICILIO VARCHAR(30));
```

Uso el comando para modificar una tabla
alter table

```
ALTER TABLE PACIENTE ADD PRIMARY KEY(DNI_PACIENTE);
```

Si debo crear una tabla que posee **primary key** compuesta y dos **foreign key** lo hago de la siguiente manera:

```
CREATE TABLE CITA ( DNI_PACIENTE BIGINT,  
                    ID_MEDICO INT,  
                    FECHA DATE,  
                    PRIMARY KEY (DNI_PACIENTE, ID_MEDICO),  
                    FOREIGN KEY (DNI_PACIENTE) REFERENCES PACIENTE(DNI_PACIENTE),  
                    FOREIGN KEY (ID_MEDICO) REFERENCES MEDICO(ID_MEDICO));
```

¿Qué más puedo hacer con el *alter table*?

→ Modificar el tipo de dato de una columna

```
ALTER TABLE PACIENTE MODIFY COLUMN EDAD FLOAT;
```

→ Modificar el nombre de una columna

```
ALTER TABLE CITA RENAME COLUMN FECHA TO FECHA_CITA;
```

→ Agregar nueva columna

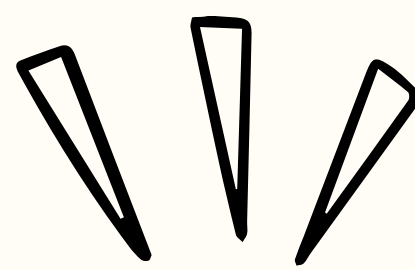
```
ALTER TABLE MEDICO ADD MAIL VARCHAR(50);
```

→ Eliminar una columna

```
ALTER TABLE PACIENTE DROP COLUMN DOMICILIO;
```



INSERCIÓN DE DATOS EN TABLAS



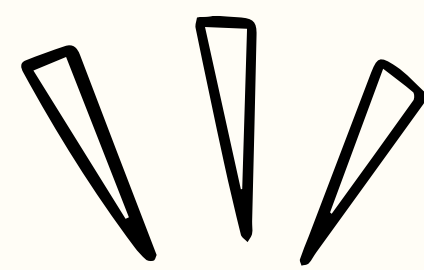
Para insertar datos en una tabla debemos usar el comando *insert into*

```
INSERT INTO CONSULTORIO (NOMBRE, DIRECCION, ID_CONS)  
VALUES ("ARCOIRIS", "CORRIENTES 700", 1);
```

Debo colocar los datos que ingresaré en el orden que detalle las columnas entre paréntesis

Si no, debo respetar el orden de cómo cree la tabla

```
INSERT INTO CONSULTORIO VALUES (2, "NUBES", "CORRIENTES 702");
```

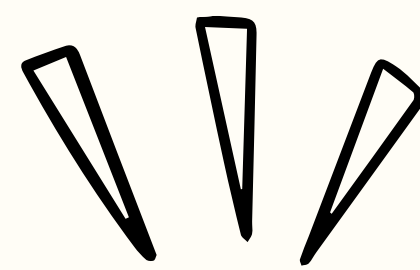


De esta manera ingresamos varios registros en la misma query

```
INSERT INTO CONSULTORIO VALUES (3, "ARBOL", "CORRIENTES 704"),  
                                  (4, "CACTUS", "CORRIENTES 706"),  
                                  (5, "AGUA", "CORRIENTES 708");
```

Los registros se ven así en la tabla

id_cons	nombre	direccion
1	Arcoiris	Corrientes 700
2	Nubes	Corrientes 702
3	Arbol	Corrientes 704
4	Cactus	Corrientes 706



Para ingresar datos en una tabla que posee FK, ese dato ya debe existir en la otra tabla

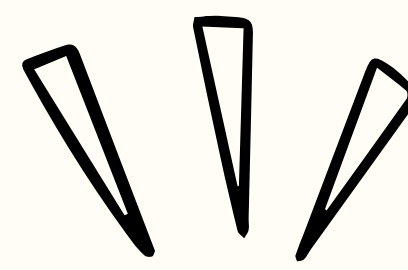
INSERT INTO MEDICO VALUES (256, "RENÉ FAVALORO", "CARDIÓLOGO", "4", "RENE@MAIL.COM");

id_cons	nombre	direccion
1	Arcoiris	Corrientes 700
2	Nubes	Corrientes 702
3	Arbol	Corrientes 704
4	Cactus	Corrientes 706

id_medico	nombre	especialidad	id_consultorio_fk	mail
256	René Favalaro	Cardiólogo	4	rene@mail.com

PRIMERAS QUERYS





Para ver todos los registros de una tabla uso

select * from tabla

SELECT * FROM CONSULTORIO;

	id_cons	nombre	direccion
▶	1	Arcoiris	Corrientes 700
	2	Nubes	Corrientes 702
	3	Arbol	Corrientes 704
	4	Cactus	Corrientes 706

Si quiero ver sólo alguna columna en específico lo debo aclarar luego del select

SELECT NOMBRE, ESPECIALIDAD, MAIL FROM MEDICO;

	nombre	especialidad	mail
▶	René Favalaro	Cardiólogo	rene@mail.com

Dentro de la query puedo poner condiciones con el **where**

```
SELECT * FROM CONSULTORIO WHERE ID_CONS = 3;
```

	id_cons	nombre	direccion
▶	3	Arbol	Corrientes 704

Los operadores que puedo usar son **=** , **<** , **>** , **<=** , **>=**

```
SELECT * FROM CONSULTORIO WHERE ID_CONS <= 3;
```

	id_cons	nombre	direccion
	1	Arcoiris	Corrientes 700
	2	Nubes	Corrientes 702
	3	Arbol	Corrientes 704

Dentro de la query puedo poner VARIAS condiciones en el where

```
SELECT * FROM CONSULTORIO WHERE ID_CONS >= 2 AND ID_CONS <=4;
```

id_cons	nombre	direccion
2	Nubes	Corrientes 702
3	Arbol	Corrientes 704
4	Cactus	Corrientes 706

```
SELECT * FROM CONSULTORIO WHERE ID_CONS >= 2 OR ID_CONS <=4;
```

id_cons	nombre	direccion
1	Arcoiris	Corrientes 700
2	Nubes	Corrientes 702
3	Arbol	Corrientes 704
4	Cactus	Corrientes 706

SELECT * FROM CONSULTORIO WHERE NOMBRE = 'ARCOIRIS' AND ID_CONS = 1;

id_cons	nombre	direccion
1	Arcoiris	Corrientes 700

SELECT * FROM CONSULTORIO WHERE NOMBRE = 'ARCOIRIS' OR ID_CONS = 4;

id_cons	nombre	direccion
1	Arcoiris	Corrientes 700
4	Cactus	Corrientes 706

al poner = lo que esta a su derecha debe coincidir exacto

Si quiero que el filtro coincida con solo una parte uso el comando **like**
me ayudo con el símbolo **%**

→ Datos que el nombre empiece con 'A'

```
SELECT * FROM CONSULTORIO WHERE NOMBRE LIKE 'A%';
```

id_cons	nombre	direccion
1	Arcoiris	Corrientes 700
3	Arbol	Corrientes 704

→ Datos que el nombre contenga 'u'

```
SELECT * FROM CONSULTORIO WHERE NOMBRE LIKE '%U%';
```

id_cons	nombre	direccion
2	Nubes	Corrientes 702
4	Cactus	Corrientes 706

→ Datos que el nombre finalice con 's'

```
SELECT * FROM CONSULTORIO WHERE NOMBRE LIKE '%S';
```

id_cons	nombre	direccion
1	Arcoiris	Corrientes 700
2	Nubes	Corrientes 702
4	Cactus	Corrientes 706

También puedo indicar en función de la cantidad de caracteres que tiene un dato para cada caracter usamos un `_`

→ Datos el nombre tenga 5 caracteres

```
SELECT * FROM CONSULTORIO WHERE NOMBRE LIKE '_____';
```

id_cons	nombre	direccion
2	Nubes	Corrientes 702
3	Arbol	Corrientes 704

Puedo indicar en qué posición tiene que estar determinado caracter

→ Datos que la tercera letra de su nombre sea 'c'

```
SELECT * FROM CONSULTORIO WHERE NOMBRE LIKE '__c%';
```

id_cons	nombre	direccion
1	Arcoiris	Corrientes 700
4	Cactus	Corrientes 706

Negaciones dentro de la query usando <> y NOT LIKE

→Mostrar los consultorios cuyo id sea *distinto a 3*

```
SELECT * FROM CONSULTORIO WHERE ID_CONS <> 3;
```

	id_cons	nombre	direccion
▶	1	Arcoiris	Corrientes 700
	2	Nubes	Corrientes 702
	4	Cactus	Corrientes 706

→Mostrar los consultorios cuyo nombre NO contenga 'a'

```
SELECT * FROM CONSULTORIO WHERE NOMBRE NOT LIKE '%A%';
```

id_cons	nombre	direccion
2	Nubes	Corrientes 702

Ordenar los dato con **ORDER BY**

→ASC (De menor a mayor o de la A a la Z)

```
SELECT * FROM CONSULTORIO ORDER BY ID_CONS ASC;
```

id_cons	nombre	direccion
1	Arcoiris	Corrientes 700
2	Nubes	Corrientes 702
3	Arbol	Corrientes 704
4	Cactus	Corrientes 706

→DESC (De mayor a menor o de la Z a la A)

```
SELECT * FROM CONSULTORIO ORDER BY NOMBRE DESC;
```

id_cons	nombre	direccion
2	Nubes	Corrientes 702
4	Cactus	Corrientes 706
1	Arcoiris	Corrientes 700
3	Arbol	Corrientes 704

Para traer registros que se encuentren entre dos valores podemos usar ***between***

```
SELECT LEGAJO, NOMBRE FROM ALUMNO  
WHERE LEGAJO BETWEEN 2 AND 4;
```

legajo	nombre
2	Arturo Frondizi
3	Hipolito Yrigoyen
4	Juan Domingo Peron

IS NULL / IS NOT NULL

Con esta condición indico que me traiga los registros que tienen determinado campo nulo o no nulo

estos son todos los registros de la tabla

LEGAJO	NOMBRE	CIUDAD
1	Jose de San Martin	Yapeyu
2	Cornelio Saavedra	Otuyo
3	Manuel Belgrano	NULL
4	Santiago de Liniers	Niort
5	Juan Martin de Pueyredon	Buenos Aires

→ Registros donde ciudad NO sea nulo

```
SELECT * FROM DOCENTE WHERE CIUDAD IS NOT NULL;
```

LEGAJO	NOMBRE	CIUDAD
1	Jose de San Martin	Yapeyu
2	Cornelio Saavedra	Otuyo
4	Santiago de Liniers	Niort
5	Juan Martin de Pueyredon	Buenos Aires

→ Registros donde ciudad SI sea nulo

```
SELECT * FROM DOCENTE WHERE CIUDAD IS NULL;
```

LEGAJO	NOMBRE	CIUDAD
3	Manuel Belgrano	NULL

IN / NOT IN

Con esta condición indico que un valor se encuentre o no se encuentra en la tabla

→ Enumerando datos, defino valores explícitos

```
SELECT * FROM ALUMNO WHERE LEGAJO IN (1, 3, 5);
```

LEGAJO	NOMBRE	CIUDAD
1	Arturo Illia	Pergamino
3	Hipolito Yrigoyen	Buenos Aires
5	Bartolome Mitre	Buenos Aires

→ Subconsulta, pido registros que se encuentren dentro de otra lista armada con una consulta

Legajos que se encuentran en la tabla cursa

```
SELECT * FROM ALUMNO WHERE LEGAJO IN (SELECT LEGAJO FROM CURSA);
```

LEGAJO	NOMBRE	CIUDAD
1	Arturo Illia	Pergamino
2	Arturo Frondizi	Paso de los Libres
4	Juan Domingo Peron	Lobos
5	Bartolome Mitre	Buenos Aires

Funciones Agregadas

Las funciones de agregación son funciones que toman una colección de valores de entrada y devuelve un solo valor

Algunas funciones disponibles son:


- AVG (promedio)
- MIN
- MAX
- SUM
- COUNT

AVG (promedio)

→ Lo uso para calcular el promedio de algo.
Los datos de entrada solo pueden ser números

por ejemplo: Promedio de nota de todos los alumnos

```
SELECT AVG(CURSA.NOTA) FROM CURSA;
```



avg(curso.nota)
8.6429

MIN (mínimo)

→ Lo uso para calcular el valor mínimo de un conjunto de valores.

por ejemplo: Obtener el menor legajo de los alumnos

```
SELECT MIN(LEGAJO) FROM ALUMNO;
```

	min(legajo)
	1

MAX (máximo)

→ Lo uso para calcular el valor máximo de un conjunto de valores.

por ejemplo: Obtener el mayor código de materia

```
SELECT MAX(CODIGO) FROM MATERIA;
```

<code>max(codigo)</code>
6

SUM (suma)

→ Lo uso para calcular la suma de determinados valores.

por ejemplo: Obtener la suma de todas las notas que se encuentren en la tabla cursa

```
SELECT SUM(NOTA) FROM CURSA;
```

sum(nota)
121

COUNT (contar)

→ Lo uso para contar cuántos valores hay.

por ejemplo: Obtener cuántos docentes tiene la escuela

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

count(*)
5

Por defecto las funciones se aplican a todas las tuplas resultantes de la consulta. Pero tenemos la posibilidad de agrupar las tuplas resultantes para poder aplicar las funciones de columna a grupos específicos usando la clausula **GROUP BY**

Debo indicar a partir de qué campo agruparé

→ Mostrar el promedio de nota por alumno

```
SELECT LEGAJO, AVG(NOTA) FROM CURSA  
GROUP BY LEGAJO;
```

legajo	avg(nota)
1	8.0000
2	9.3333
4	8.3333
5	10.0000

Adicionalmente podemos aplicar condiciones sobre los grupos utilizando la clausula **HAVING**

Having me permite ponerle una condición que se tiene que cumplir para que funcione o filtre ese group by

→ Mostrar el promedio de nota por alumno siendo el promedio mayor a 8

```
SELECT LEGAJO, AVG(NOTA) FROM CURSA  
GROUP BY LEGAJO HAVING AVG(NOTA) > 8;
```

legajo	avg(nota)
2	9.3333
4	8.3333
5	10.0000

Cuando usemos funciones agregadas lo más probable es que la sentencia se conforme así

SELECT + FUNC_AG() + FROM + WHERE + GROUP BY + HAVING

SELF JOIN

ID_EMP	USERNAME	NOMBRE	APELLIDO	TELEFONO	SUPERIOR
1	juanperez	Juan	Perez	1234567890	NULL
2	marialopez	Maria	Lopez	0987654321	1
3	carlossoto	Carlos	Soto	1122334455	1
4	luciafernandez	Lucia	Fernandez	2233445566	2
5	danielgarcia	Daniel	Garcia	3344556677	2
6	anamoreno	Ana	Moreno	4455667788	3
7	pedroramirez	Pedro	Ramirez	5566778899	4
8	sofiatorres	Sofia	Torres	6677889900	4
9	pablogomez	Pablo	Gomez	7788990011	5
10	marcelodominguez	Marcelo	Dominguez	8899001122	6
11	clararodriguez	Clara	Rodriguez	9900112233	6
12	fabianmartinez	Fabian	Martinez	1011121314	NULL
13	mariaflores	Maria	Flores	1213141516	7
14	gustavopereira	Gustavo	Pereira	1415161718	8

Uno una tabla con ella misma para ver el nombre de su superior

```
SELECT EMP1.ID_EMP,  
EMP1.USERNAME,  
EMP1.NOMBRE,  
EMP1.SUPERIOR,  
EMP2.NOMBRE  
NOMBRE_SUP  
FROM EMPLEADO EMP1  
LEFT JOIN EMPLEADO EMP2  
ON EMP1.SUPERIOR =  
EMP2.ID_EMP;
```

ID_EMP	USERNAME	NOMBRE	SUPERIOR	NOMBRE_SUP
1	juanperez	Juan	NULL	NULL
2	marialopez	Maria	1	Juan
3	carlossoto	Carlos	1	Juan
4	luciafernandez	Lucia	2	Maria
5	danielgarcia	Daniel	2	Maria
6	anamoreno	Ana	3	Carlos
7	pedroramirez	Pedro	4	Lucia
8	sofiatorres	Sofia	4	Lucia
9	pablogomez	Pablo	5	Daniel
10	marcelodominguez	Marcelo	6	Ana
11	clararodriguez	Clara	6	Ana
12	fabianmartinez	Fabian	NULL	NULL
13	mariaflores	Maria	7	Pedro
14	gustavopereira	Gustavo	8	Sofia
15	fernandocastro	Fernando	8	Sofia

Para traer registros que se encuentren
entre dos valores podemos usar
between

```
SELECT LEGAJO, NOMBRE FROM ALUMNO  
WHERE LEGAJO BETWEEN 2 AND 4;
```

legajo	nombre
2	Arturo Frondizi
3	Hipolito Yrigoyen
4	Juan Domingo Peron

DISTINCT

SELECT LEGAJO FROM CURSA ORDER BY LEGAJO;

LEGAJO
1
1
1
1
1
1
2
2
2
4
4
4
5
5

SELECT DISTINCT LEGAJO FROM CURSA ORDER BY LEGAJO;

LEGAJO
1
2
4
5

IS NULL / IS NOT NULL

Con esta condición indico que me traiga los registros que tienen determinado campo nulo o no nulo

estos son todos los registros de la tabla

LEGAJO	NOMBRE	CIUDAD
1	Jose de San Martin	Yapeyu
2	Cornelio Saavedra	Otuyo
3	Manuel Belgrano	NULL
4	Santiago de Liniers	Niort
5	Juan Martin de Pueyredon	Buenos Aires

→ Registros donde ciudad NO sea nulo

```
SELECT * FROM DOCENTE WHERE CIUDAD IS NOT NULL;
```

LEGAJO	NOMBRE	CIUDAD
1	Jose de San Martin	Yapeyu
2	Cornelio Saavedra	Otuyo
4	Santiago de Liniers	Niort
5	Juan Martin de Pueyredon	Buenos Aires

→ Registros donde ciudad SI sea nulo

```
SELECT * FROM DOCENTE WHERE CIUDAD IS NULL;
```

LEGAJO	NOMBRE	CIUDAD
3	Manuel Belgrano	NULL

IN / NOT IN

Con esta condición indico que un valor se encuentre o no se encuentra en la tabla

→ Enumerando datos, defino valores explícitos

```
SELECT * FROM ALUMNO WHERE LEGAJO IN (1, 3, 5);
```

LEGAJO	NOMBRE	CIUDAD
1	Arturo Illia	Pergamino
3	Hipolito Yrigoyen	Buenos Aires
5	Bartolome Mitre	Buenos Aires

→ Subconsulta, pido registros que se encuentren dentro de otra lista armada con una consulta

Legajos que se encuentran en la tabla cursa

```
SELECT * FROM ALUMNO WHERE LEGAJO IN (SELECT LEGAJO FROM CURSA);
```

LEGAJO	NOMBRE	CIUDAD
1	Arturo Illia	Pergamino
2	Arturo Frondizi	Paso de los Libres
4	Juan Domingo Peron	Lobos
5	Bartolome Mitre	Buenos Aires

EXISTS/ NOT EXISTS

Con esta condición indico que un valor se encuentre o no se encuentra en la tabla

→ Alumnos que cursan materias

```
SELECT * FROM ALUMNO WHERE EXISTS (SELECT LEGAJO FROM CURSA WHERE ALUMNO.LEGAJO = CURSA.LEGAJO);
```

LEGAJO	NOMBRE	CIUDAD
1	Arturo Illia	Pergamino
2	Arturo Frondizi	Paso de los Libres
4	Juan Domingo Peron	Lobos
5	Bartolome Mitre	Buenos Aires

→ Alumnos que no cursan materias

```
SELECT * FROM ALUMNO WHERE NOT EXISTS (SELECT * FROM CURSA WHERE ALUMNO.LEGAJO = CURSA.LEGAJO);
```

LEGAJO	NOMBRE	CIUDAD
3	Hipolito Yrigoyen	Buenos Aires




JOIN → Consulta entre múltiples tablas

INNER JOIN: Devuelve las filas donde hay coincidencia en cierto campo de ambas tablas

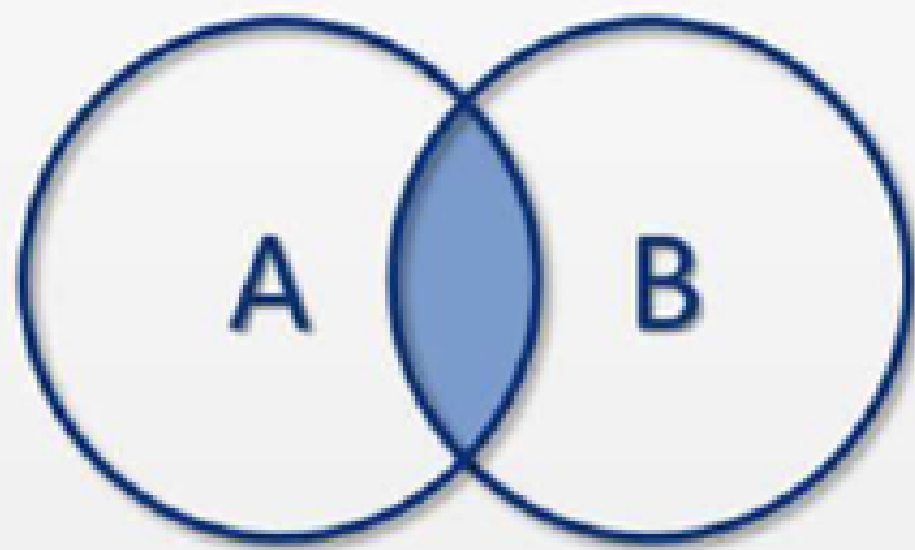
LEFT JOIN: Devuelve todas las filas de la tabla de la izquierda del left join + todos los registros en los que encuentre coincidencia de la tabla de la derecha

RIGHT JOIN: Devuelve todas las filas de la tabla de la derecha del right join + todos los registros en los que encuentre coincidencia de la tabla de la izquierda

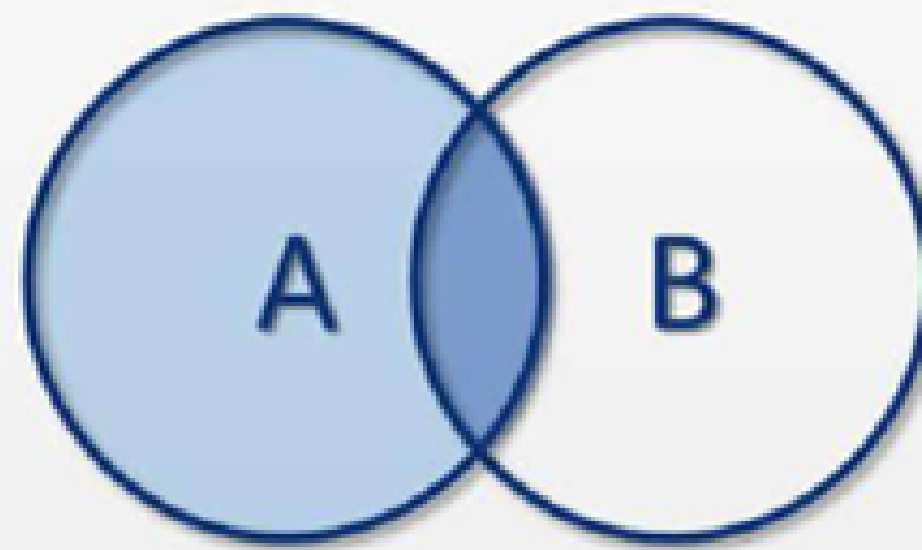
NATURAL JOIN: Devuelve las filas donde hay coincidencia en el campo de igual nombre de ambas tablas



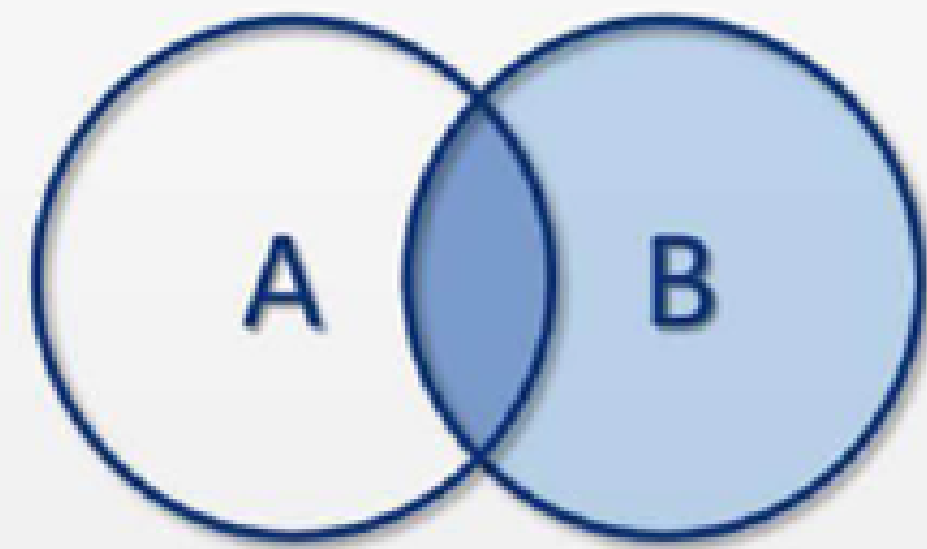
INNER JOIN



OUTER JOIN



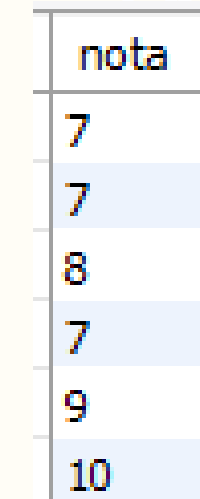
LEFT



RIGHT

INNER JOIN: Listar las notas del alumno Arturo Illia

```
SELECT CURSA.NOTA FROM ALUMNO INNER JOIN CURSA
ON ALUMNO.LEGAJO = CURSA.LEGAJO
WHERE ALUMNO.NOMBRE = 'ARTURO ILLIA'
```



nota
7
7
8
7
9
10

NATURAL JOIN: Listar nombre, legajo y código de materia que cursa Arturo Frondizi

```
SELECT ALUMNO.NOMBRE, ALUMNO.LEGAJO, CURSA.CODIGO
FROM ALUMNO NATURAL JOIN CURSA
WHERE ALUMNO.NOMBRE = 'ARTURO FRONDIZI';
```

nombre	legajo	codigo
Arturo Frondizi	2	1
Arturo Frondizi	2	3
Arturo Frondizi	2	5

LEFT JOIN: Listar todos los alumnos registrados y el código y nota de la materia en el caso que cursen

```
SELECT A.NOMBRE, A.LEGAJO, C.CODIGO, C.NOTA
FROM ALUMNO A LEFT JOIN CURSA C
ON A.LEGAJO = C.LEGAJO
```

nombre	legajo	codigo	nota
Arturo Illia	1	1	7
Arturo Illia	1	2	7
Arturo Illia	1	3	8
Arturo Illia	1	4	7
Arturo Illia	1	5	9
Arturo Illia	1	6	10
Arturo Frondizi	2	1	10
Arturo Frondizi	2	3	9
Arturo Frondizi	2	5	9
Hipolito Yrigoyen	3	NULL	NULL
Juan Domingo P...	4	2	9
Juan Domingo P...	4	4	7
Juan Domingo P...	4	6	9
Bartolome Mitre	5	1	10
Bartolome Mitre	5	5	10

RIGHT JOIN: Listar todas los códigos de notas que son cursados por alumnos, de ellos el nombre y legajo

```
SELECT A.NOMBRE, A.LEGAJO, C.CODIGO
FROM ALUMNO A RIGHT JOIN CURSA C
ON A.LEGAJO = C.LEGAJO;
```

	nombre	legajo	codigo
►	Arturo Illia	1	1
	Arturo Frondizi	2	1
	Bartolome Mitre	5	1
	Arturo Illia	1	2
	Juan Domingo Peron	4	2
	Arturo Illia	1	3
	Arturo Frondizi	2	3
	Arturo Illia	1	4
	Juan Domingo Peron	4	4
	Arturo Illia	1	5
	Arturo Frondizi	2	5
	Bartolome Mitre	5	5
	Arturo Illia	1	6
	Juan Domingo Peron	4	6