

# </ Procesual Hito 3 Base de Datos I - 2023

/>



Bases de Datos - SQL  
Server

by Jamil Alex Quispe  
Valencia

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1

# </ Consigna Y Detalle del problema

- Diseñe un sistema de Base de Datos Relacional utilizando el gestor de Base de Datos SQL Server teniendo como premisa el uso de buenas prácticas en diseño de la base de datos aplicados al siguiente escenario. Una comunidad de estudiantes de la nación UNIFRANZ de nombre los UNIFRANZITOS desea implementar un nuevo sistema para poder administrar los CAMPEONATOS DE FÚTBOL de todas las sedes. Es decir crear un campeonato en donde puedan participar todas las sedes, en el campeonato pueden inscribirse tanto categoría varones y mujeres.
- Detalle del problema: Se tiene como contexto un CAMPEONATO DE FÚTBOL en el cual se tiene 3 entidades principales el campeonato como tal, los equipos que participaran en el campeonato y en donde cada equipo tendrá una cantidad de jugadores. En tal sentido se deberá crear las siguientes tablas
  - campeonato
  - equipo
  - jugador

# </ Detalle de las tablas.

## campeonato

**id\_campeonato** => cadena de 12 caracteres y ademas llave primaria  
**nombre\_campeonato** => una cadena de 30 caracteres que no acepta valores nulos  
**sede** => una cadena de 20 caracteres que no acepta valores nulos

## equipo

**id\_equipo** => cadena de 12 caracteres y ademas llave primaria  
**nombre\_equipo** => una cadena de 30 caracteres, que no acepta valores nulos  
**categoria** => esta columna recibe valores como (varones o mujeres), que no acepta valores nulos  
**id\_campeonato** => llave foreign key relacionado con la tabla **campeonato**

## jugador

**id\_jugador** => cadena de 12 caracteres y ademas llave primaria  
**nombres** => una cadena de 30 caracteres, que no acepta valores nulos  
**apellidos** => una cadena de 50 caracteres, que no acepta valores nulos  
**ci** => una cadena de 15 caracteres (ejem: 8997899LP), que no acepta valores nulos  
**edad** => un valor numérico, que no acepta valores nulos  
**id\_equipo** => llave foreign key relacionado con la tabla **equipo**

# </ 1.- Diseñamos nuestra propia base de datos con los Detalles y consignas asignadas:

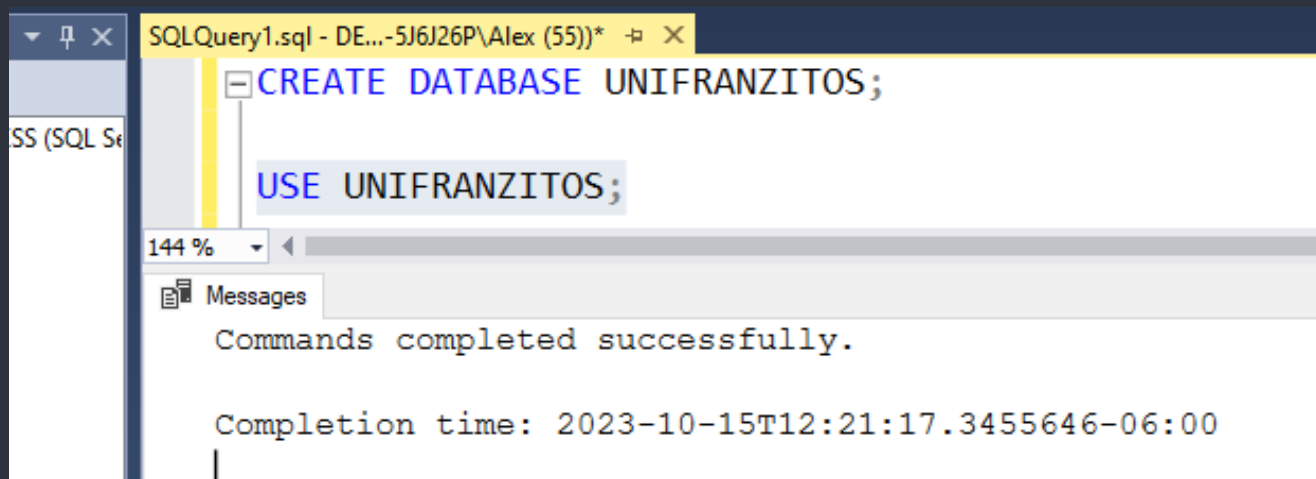
Creamos una base de datos con el nombre "UNIFRANZITOS"



The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the 'Server Enterprise' tree structure. The right pane shows the 'SQLQuery1.sql' file with the following SQL code:

```
CREATE DATABASE UNIFRANZITOS;  
  
USE UNIFRANZITOS;
```

Selección de la base de datos recién creada: la base de datos “UNIFRANZITOS”



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the SQL query: `CREATE DATABASE UNIFRANZITOS;` followed by `USE UNIFRANZITOS;`. The bottom pane, titled "Messages", shows the output: "Commands completed successfully." and "Completion time: 2023-10-15T12:21:17.3455646-06:00".

```
SQLQuery1.sql - DE...-5J6J26P\Alex (55))* X
```

```
CREATE DATABASE UNIFRANZITOS;
```

```
USE UNIFRANZITOS;
```

144 %

Messages

Commands completed successfully.

Completion time: 2023-10-15T12:21:17.3455646-06:00





Creamos la tabla campeonato con tres columnas: id\_campeonato, nombre\_campeonato y sede. La columna id\_campeonato es la clave primaria.



```
CREATE TABLE campeonato
(
    id_campeonato VARCHAR(12) PRIMARY KEY,
    nombre_campeonato VARCHAR(30) NOT NULL,
    sede VARCHAR(20) NOT NULL
);
```

Diseño a replicar de la tabla

campeonato		
	id_campeonato	varchar(12)
	nombre_campeonato	varchar(30)
	sede	varchar(20)



Insertamos datos en la tabla campeonato: Se insertan dos registros en la tabla campeonato con información sobre campeonatos, especificando el id\_campeonato, el nombre\_campeonato y la sede.



```
INSERT INTO campeonato (id_campeonato, nombre_campeonato, sede)
VALUES ('camp-111', 'Campeonato Unifranz', 'El Alto');
INSERT INTO campeonato (id_campeonato, nombre_campeonato, sede)
VALUES ('camp-222', 'Campeonato Unifranz', 'Cochabamba');

SELECT * FROM campeonato;
```



Creemos la tabla equipo: Se crea la tabla equipo con cinco columnas: id\_equipo, nombre\_equipo, categoria, id\_campeonato, y una clave [FOREIGN KEY] que está relacionada con la tabla campeonato a través del campo id\_campeonato. La columna id\_equipo es la clave primaria.



Diseño a replicar de la tabla

```
CREATE TABLE equipo
(
    id_equipo VARCHAR(12) PRIMARY KEY,
    nombre_equipo VARCHAR(30) NOT NULL,
    categoria VARCHAR(8) NOT NULL ,
    id_campeonato VARCHAR(12),
    FOREIGN KEY (id_campeonato) REFERENCES campeonato (id_campeonato)
);
```

equipo	
id_equipo	varchar(12)
nombre_equipo	varchar(30)
categoria	varchar(8)
id_campeonato	varchar(12)





Insertamos datos tabla equipo: Se insertan tres registros en la tabla equipo con información sobre equipos, especificando el id\_equipo, el nombre\_equipo, la categoria y el id\_campeonato al que pertenecen.



```
INSERT INTO equipo (id_equipo, nombre_equipo, categoria, id_campeonato)
VALUES ('equ-111', 'Google', 'VARONES', 'camp-111'),
('equ-222', '404 Not found', 'VARONES', 'camp-111'),
('equ-333', 'girls unifranz', 'MUJERES', 'camp-111');
```

```
SELECT * FROM equipo
```



Creamos la tabla jugador: Se crea la tabla jugador con seis columnas: id\_jugador, nombres, apellidos, ci, edad y una clave foránea que está relacionada con la tabla equipo a través del campo id\_equipo. La columna id\_jugador es la clave primaria.



SQLQuery1.sql - DE...-5J6J26P\Alex (55))

```
CREATE TABLE jugador
(
    id_jugador VARCHAR(12) PRIMARY KEY,
    nombres VARCHAR(30) NOT NULL,
    apellidos VARCHAR(50) NOT NULL,
    ci VARCHAR(15) NOT NULL,
    edad INT NOT NULL,
    id_equipo VARCHAR(12),
    FOREIGN KEY (id_equipo) REFERENCES equipo (id_equipo)
);
```

Diseño a replicar de la tabla

jugador	
id_jugador	varchar(12)
nombres	varchar(30)
apellidos	varchar(50)
ci	varchar(15)
edad	int
id_equipo	varchar(12)

Insertamos en la tabla jugador: Se insertan cinco registros en la tabla jugador con información sobre jugadores, especificando el id\_jugador, nombres, apellidos, ci, edad y el id\_equipo al que pertenecen.



```
);  
  
INSERT INTO jugador (id_jugador, nombres, apellidos, ci, edad, id_equipo )  
VALUES ('jug-111', 'Carlos', 'Villa', '8997811LP', 19, 'equ-222'),  
      ('jug-222', 'Pedro', 'Salas', '8997822LP', 20, 'equ-222'),  
      ('jug-333', 'Saul', 'Araja', '8997833LP', 21, 'equ-222'),  
      ('jug-444', 'Sandra', 'Solis', '8997844LP', 20, 'equ-333'),  
      ('jug-555', 'Ana', 'Mica', '8997855LP', 23, 'equ-333');  
  
SELECT * FROM jugador
```



</ Con el comando `SELECT*FROM` mostramos todas las tablas creadas

```
--MOSTRAMOS TODAS LAS TABLAS
```

```
SELECT * FROM campeonato
```

```
SELECT* FROM equipo
```

```
SELECT* FROM jugador
```

</ Comparamos si los datos ingresados son los mismos que deben estar ingresados. TABLA campeonato

```
SELECT * FROM campeonato;
```

158 %

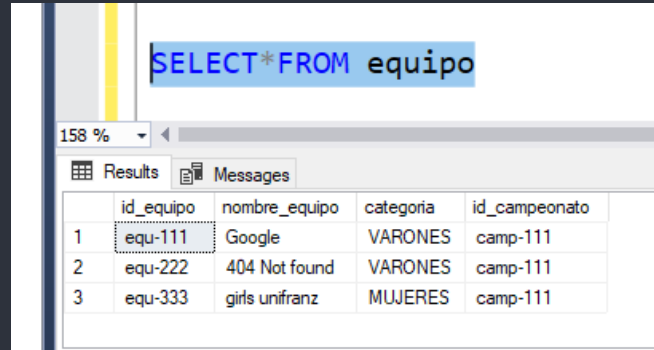
Results Messages

	id_campeonato	nombre_campeonato	sede
1	camp-111	Campeonato Unifranz	El Alto
2	camp-222	Campeonato Unifranz	Cochabamba

tabla campeonato

id_campeonato	nombre_campeonato	sede
camp-111	Campeonato Unifranz	El Alto
camp-222	Campeonato Unifranz	Cochabamba

</ Comparamos si los datos ingresados son los mismos que deben estar ingresados. TABLA equipo



The screenshot shows a database query interface. At the top, the SQL command `SELECT * FROM equipo` is entered in a text box. Below the text box, there is a zoom level of 158% and two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with the following data:

	id_equipo	nombre_equipo	categoria	id_campeonato
1	equ-111	Google	VARONES	camp-111
2	equ-222	404 Not found	VARONES	camp-111
3	equ-333	girls unifranz	MUJERES	camp-111

**tabla equipo**

id_equipo	nombre_equipo	categoria	id_campeonato
equ-111	Google	VARONES	camp-111
equ-222	404 Not found	VARONES	camp-111
equ-333	girls unifranz	MUJERES	camp-111

</ Comparamos si los datos ingresados son los mismos que deben estar ingresados. TABLA jugador

SELECT \* FROM jugador

158 %

Results Messages

	id_jugador	nombres	apellidos	ci	edad	id_equipo
1	jug-111	Carlos	Villa	8997811LP	19	equ-222
2	jug-222	Pedro	Salas	8997822LP	20	equ-222
3	jug-333	Saul	Araj	8997833LP	21	equ-222
4	jug-444	Sandra	Solis	8997844LP	20	equ-333
5	jug-555	Ana	Mica	8997855LP	23	equ-333

tabla jugador

id_jugador	nombres	apellidos	ci	edad	id_equipo
jug-111	Carlos	Villa	8997811LP	19	equ-222
jug-222	Pedro	Salas	8997822LP	20	equ-222
jug-333	Saul	Araj	8997833LP	21	equ-222
jug-444	Sandra	Solis	8997844LP	20	equ-333
jug-555	Ana	Mica	8997855LP	23	equ-333

Activar Windows  
Ve a Configuración para activar W

</ Realizamos una comparacion general

tabla campeonato			
id_campeonato	nombre_campeonato	sede	
camp-111	Campeonato Unifranz	El Alto	
camp-222	Campeonato Unifranz	Cochabamba	

tabla equipo			
id_equipo	nombre_equipo	categoria	id_campeonato
equ-111	Google	VARONES	camp-111
equ-222	404 Not found	VARONES	camp-111
equ-333	girls unifranz	MUJERES	camp-111

tabla jugador					
id_jugador	nombres	apellidos	ci	edad	id_equipo
jug-111	Carlos	Villa	8997811LP	19	equ-222
jug-222	Pedro	Salas	8997822LP	20	equ-222
jug-333	Saul	Araj	8997833LP	21	equ-222
jug-444	Sandra	Solis	8997844LP	20	equ-333
jug-555	Ana	Mica	8997855LP	23	equ-333

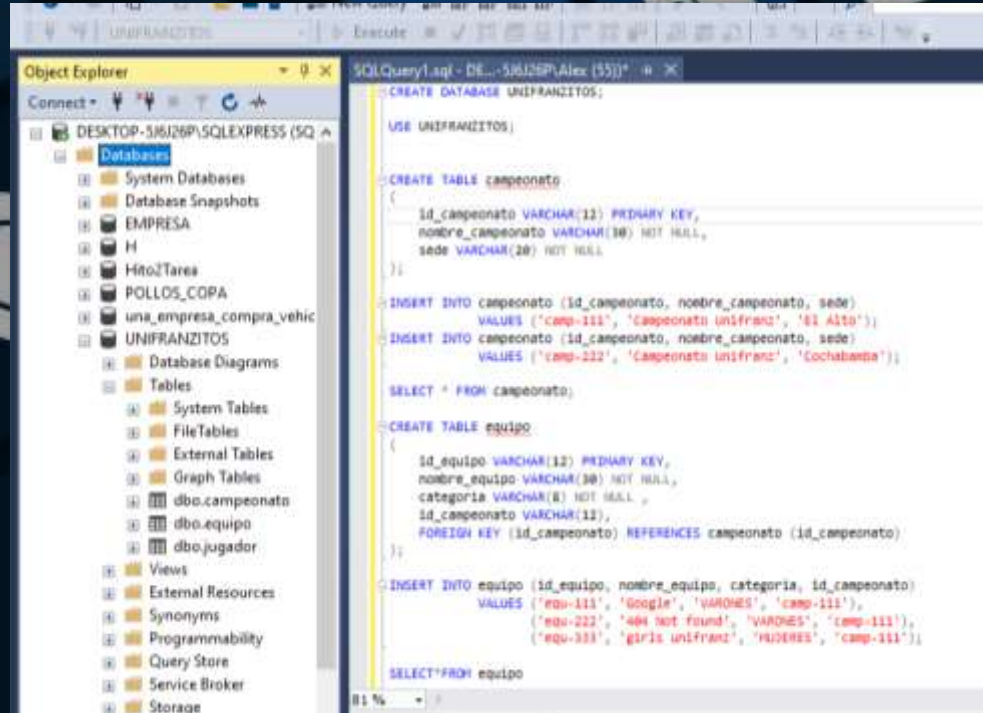
Results					
Messages					
	id_campeonato	nombre_campeonato	sede		
1	camp-111	Campeonato Unifranz	El Alto		
2	camp-222	Campeonato Unifranz	Cochabamba		

	id_equipo	nombre_equipo	categoria	id_campeonato	
1	equ-111	Google	VARONES	camp-111	
2	equ-222	404 Not found	VARONES	camp-111	
3	equ-333	girls unifranz	MUJERES	camp-111	

	id_jugador	nombres	apellidos	ci	edad	id_equipo
1	jug-111	Carlos	Villa	8997811LP	19	equ-222
2	jug-222	Pedro	Salas	8997822LP	20	equ-222
3	jug-333	Saul	Araj	8997833LP	21	equ-222
4	jug-444	Sandra	Solis	8997844LP	20	equ-333
5	jug-555	Ana	Mica	8997855LP	23	equ-333



## </ ||.- Manejo de conceptos



The screenshot displays the SQL Server Enterprise Manager interface on the left and the SQL Query Editor on the right. The Enterprise Manager shows a tree view of the 'UNIFRANZITOS' database, including tables like 'dbo.campeonato', 'dbo.equipo', and 'dbo.jugador'. The Query Editor shows the following SQL script:

```
SQLQuery1.sql - DE...-586J26P(Alex (55))

-- CREATE DATABASE UNIFRANZITOS;

USE UNIFRANZITOS;

-- CREATE TABLE campeonato
(
    id_campeonato VARCHAR(11) PRIMARY KEY,
    nombre_campeonato VARCHAR(30) NOT NULL,
    sede VARCHAR(20) NOT NULL
);

-- INSERT INTO campeonato (id_campeonato, nombre_campeonato, sede)
VALUES ('camp-111', 'Campeonato Unifranz', 'El Alto');
-- INSERT INTO campeonato (id_campeonato, nombre_campeonato, sede)
VALUES ('camp-222', 'Campeonato Unifranz', 'Cochabamba');

SELECT * FROM campeonato;

-- CREATE TABLE equipo
(
    id_equipo VARCHAR(12) PRIMARY KEY,
    nombre_equipo VARCHAR(30) NOT NULL,
    categoria VARCHAR(8) NOT NULL,
    id_campeonato VARCHAR(11),
    FOREIGN KEY (id_campeonato) REFERENCES campeonato (id_campeonato)
);

-- INSERT INTO equipo (id_equipo, nombre_equipo, categoria, id_campeonato)
VALUES ('equ-111', 'Google', 'VARONES', 'camp-111'),
('equ-222', '404 Not Found', 'VARONES', 'camp-111'),
('equ-333', 'girls unifranz', 'MUJERES', 'camp-111');

SELECT * FROM equipo
```

</Que es DDL y DML, adicionalmente muestra un ejemplo en la base de datos UNIFRANZITOS.

¿Que es DDL ? Los comandos DDL (Data Definition Language o Lenguaje de Definición de Datos) en SQL se utilizan para definir, modificar y eliminar la estructura de la base de datos. Aquí están los comandos DDL utilizados en la creación de la base de datos "UNIFRANZITOS":

```
CREATE DATABASE UNIFRANZITOS;

USE UNIFRANZITOS;

CREATE TABLE campeonato
(
    id_campeonato VARCHAR(12) PRIMARY KEY,
    nombre_campeonato VARCHAR(30) NOT NULL,
    sede VARCHAR(20) NOT NULL
);

DROP TABLE campeonato;
```

</Que es DDL y DML, adicionalmente muestra un ejemplo en la base de datos UNIFRANZITOS.

**CREATE DATABASE:** Se utiliza para crear una nueva base de datos.

```
Query1.sql - DE...-5J6J26P\Alex (52))* -# X  
CREATE DATABASE UNIFRANZITOS;
```

**CREATE TABLE:** Se utiliza para crear una nueva tabla en la base de datos.

```
Query1.sql - DE...-5J6J26P\Alex (53))* -# X  
CREATE TABLE jugador  
(  
    id_jugador VARCHAR(12) PRIMARY KEY,  
    nombres VARCHAR(30) NOT NULL,  
    apellidos VARCHAR(50) NOT NULL,  
    ci VARCHAR(15) NOT NULL,  
    edad INT NOT NULL,  
    id_equipo VARCHAR(12),  
    FOREIGN KEY (id_equipo) REFERENCES equipo (id_equipo)  
);
```

</Que es DDL y DML, adicionalmente muestra un ejemplo en la base de datos UNIFRANZITOS.

**DROP TABLE:** Se utiliza para eliminar una tabla de la base de datos.

**ALTER TABLE:** Se utiliza para modificar la estructura de una tabla existente. Ejemplos típicos incluyen agregar, modificar o eliminar columnas.

```
DROP TABLE campeonato;
```

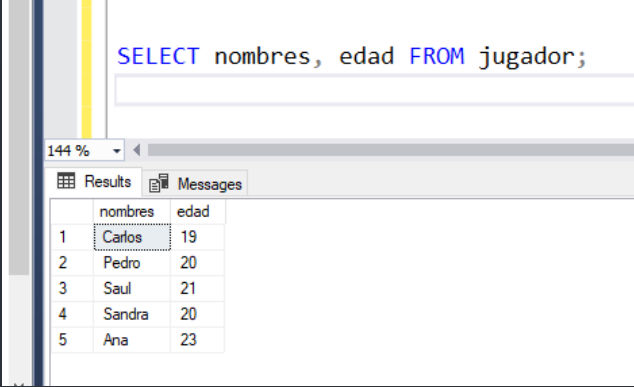
```
= ALTER TABLE equipo  
  ADD direccion VARCHAR(50);
```

</Que es DDL y DML, adicionalmente muestra un ejemplo en la base de datos UNIFRANZITOS.

¿Que es DML ? DML [Data Manipulation Language] en SQL Server es un conjunto de comandos SQL utilizados para manipular y gestionar datos en una base de datos. Los comandos DML se centran en realizar operaciones como la inserción, actualización, eliminación y consulta de datos en tablas. Los comandos DML más comunes en SQL Server son:

</Que es DDL y DML, adicionalmente muestra un ejemplo en la base de datos UNIFRANZITOS.

SELECT: Recuperar datos de la tabla "jugador".



The screenshot shows a database query window. At the top, a text box contains the SQL query: `SELECT nombres, edad FROM jugador;`. Below the text box, there are tabs for 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with two columns: 'nombres' and 'edad'. The table contains five rows of data, numbered 1 to 5 in the first column. The first row is highlighted with a dashed border.

	nombres	edad
1	Carlos	19
2	Pedro	20
3	Saul	21
4	Sandra	20
5	Ana	23

SELECT: Recuperar datos de la tabla "jugador".

```
INSERT INTO jugador (id_jugador, nombres, apellidos, ci, edad, id_equipo )
VALUES ('jug-111', 'Carlos', 'Villa', '8997811LP', 19, 'equ-222'),
('jug-222', 'Pedro', 'Salas', '8997822LP', 20, 'equ-222'),
('jug-333', 'Saul', 'Araj', '8997833LP', 21, 'equ-222'),
('jug-444', 'Sandra', 'Solis', '8997844LP', 20, 'equ-333'),
('jug-555', 'Ana', 'Mica', '8997855LP', 23, 'equ-333');
```

</Que es ODL y DML, adicionalmente muestra un ejemplo en la base de datos UNIFRANZITOS.

UPDATE: Modificar la edad de un jugador en la tabla "jugador".

```
UPDATE jugador  
SET edad = 21  
WHERE id_jugador = 'jug-111';  
  
SELECT* FROM jugador
```

174 %

Results

Messages

	id_jugador	nombres	apellidos	ci	edad	id_equipo
1	jug-111	Carlos	Villa	8997811LP	21	equ-222
2	jug-222	Pedro	Salas	8997822LP	20	equ-222
3	jug-333	Saul	Araj	8997833LP	21	equ-222

En la tabla anterior:

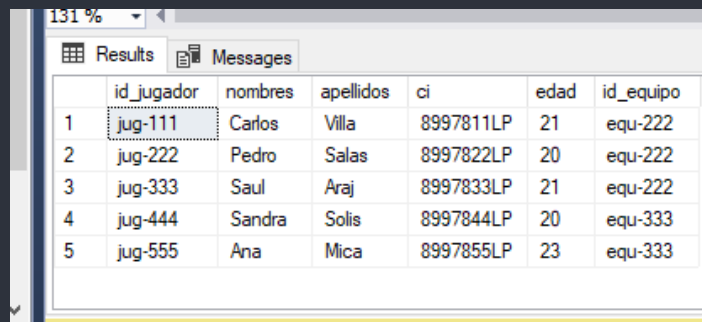
	id_jugador	nombres	apellidos	ci	edad	id_equipo
1	jug-111	Carlos	Villa	8997811LP	19	equ-222
2	jug-222	Pedro	Salas	8997822LP	20	equ-222
3	jug-333	Saul	Araj	8997833LP	21	equ-222

</Que es DDL y DML, adicionalmente muestra un ejemplo en la base de datos UNIFRANZITOS.

DELETE: Eliminar un jugador de la tabla "jugador".

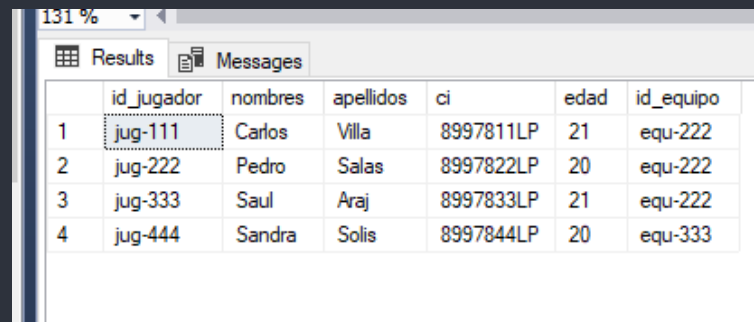
```
DELETE FROM jugador  
WHERE id_jugador = 'jug-555';  
  
SELECT* FROM jugador
```

Esta la tabla antes y después de ejecutar el comando:



131 %

	id_jugador	nombres	apellidos	ci	edad	id_equipo
1	jug-111	Carlos	Villa	8997811LP	21	equ-222
2	jug-222	Pedro	Salas	8997822LP	20	equ-222
3	jug-333	Saul	Araj	8997833LP	21	equ-222
4	jug-444	Sandra	Solis	8997844LP	20	equ-333
5	jug-555	Ana	Mica	8997855LP	23	equ-333



131 %

	id_jugador	nombres	apellidos	ci	edad	id_equipo
1	jug-111	Carlos	Villa	8997811LP	21	equ-222
2	jug-222	Pedro	Salas	8997822LP	20	equ-222
3	jug-333	Saul	Araj	8997833LP	21	equ-222
4	jug-444	Sandra	Solis	8997844LP	20	equ-333



</ Que significa PRIMARY KEY y FOREIGN KEY.

### PRIMARY KEY [Clave Primaria]:

La clave primaria es un atributo o conjunto de atributos en una tabla de una base de datos relacional que identifica de manera única cada fila o registro en esa tabla.

Las características clave de una clave primaria son:

Debe contener valores únicos, es decir, no puede haber dos filas con el mismo valor en la columna de clave primaria.

No puede contener valores nulos, lo que significa que cada fila debe tener un valor en la columna de clave primaria.

Cada tabla puede tener solo una clave primaria.

La clave primaria se utiliza para indexar y relacionar filas en una tabla con filas en otras tablas (a través de claves foráneas) para mantener la integridad referencial y permitir consultas eficientes.

PRIMARY KEY [Clave Primaria]:

```
id_campeonato VARCHAR(12) PRIMARY KEY,
```

```
id_equipo VARCHAR(12) PRIMARY KEY,
```

```
id_jugador VARCHAR(12) PRIMARY KEY,
```

## FOREIGN KEY [Clave Foránea]:

La clave foránea es un atributo o conjunto de atributos en una tabla que se relaciona con la clave primaria de otra tabla.

La clave foránea se utiliza para establecer relaciones entre tablas. Sirve para garantizar la integridad referencial, lo que significa que no se pueden insertar valores en la columna de clave foránea que no existan en la columna de clave primaria a la que se hace referencia.

Ayuda a mantener la coherencia de los datos al enlazar información en diferentes tablas.

Una tabla puede tener múltiples claves foráneas, y estas pueden referenciar la misma tabla o diferentes tablas.

Por ejemplo, si tienes una base de datos que almacena información sobre clientes y pedidos, podrías tener una tabla de "Clientes" con una clave primaria que es el número de identificación del cliente y una tabla de "Pedidos" con una clave foránea que hace referencia al número de identificación del cliente. Esto permite vincular cada pedido a un cliente específico y garantizar que solo se puedan crear pedidos para clientes que existen en la tabla de clientes, lo que mantiene la integridad de los datos.

## FOREIGN KEY [Clave Foránea]:

```
CREATE TABLE equipo
(
    id_equipo VARCHAR(12) PRIMARY KEY,
    nombre_equipo VARCHAR(30) NOT NULL,
    categoria VARCHAR(8) NOT NULL,
    id_campeonato VARCHAR(12),
    FOREIGN KEY (id_campeonato) REFERENCES campeonato (id_campeonato)
);
```

```
CREATE TABLE jugador
(
    id_jugador VARCHAR(12) PRIMARY KEY,
    nombres VARCHAR(30) NOT NULL,
    apellidos VARCHAR(50) NOT NULL,
    ci VARCHAR(15) NOT NULL,
    edad INT NOT NULL,
    id_equipo VARCHAR(12),
    FOREIGN KEY (id_equipo) REFERENCES equipo (id_equipo)
);
```

</ Defina que es una TABLA y el uso de IDENTITY.

## TABLA

Es una estructura fundamental que se utiliza para almacenar y organizar datos. Una tabla se compone de filas y columnas, donde cada fila representa un registro de datos y cada columna representa un atributo o campo de información específico. Cada celda en la tabla contiene un valor de datos único que se relaciona con una combinación de fila y columna.

A continuación, algunos aspectos clave relacionados con las tablas en bases de datos:

**Nombre de la tabla:** Cada tabla tiene un nombre único que la identifica en la base de datos.

**Columnas:** Las columnas de una tabla definen los tipos de datos que se pueden almacenar en la tabla. Cada columna tiene un nombre y un tipo de dato específico [como números enteros, cadenas de texto, fechas, etc.].

**Filas:** Las filas representan registros individuales en la tabla. Cada fila contiene valores para cada una de las columnas definidas en la tabla.

**Clave primaria:** Como mencioné anteriormente, una tabla generalmente tiene una columna o conjunto de columnas que actúan como clave primaria. La clave primaria garantiza la unicidad de cada fila en la tabla.

</ Defina que es una TABLA y el uso de IDENTITY.

## IDENTITY

El término "IDENTITY" se refiere a una propiedad o característica que se utiliza en algunos sistemas de gestión de bases de datos (DBMS), como Microsoft SQL Server, para generar valores de forma automática para una columna, por lo general, para servir como clave primaria. La columna con la propiedad IDENTITY se configura para incrementar automáticamente su valor en uno con cada nueva fila que se inserta en la tabla. Esto es útil para asignar valores únicos a registros sin la necesidad de que el usuario o la aplicación proporcione un valor específico. Algunas de las características típicas de una columna IDENTITY incluyen:

**Valores únicos y crecientes:** Cada fila tendrá un valor único y mayor que el anterior.

**Facilita la gestión de la clave primaria:** La columna con IDENTITY se usa comúnmente como clave primaria debido a su unicidad.

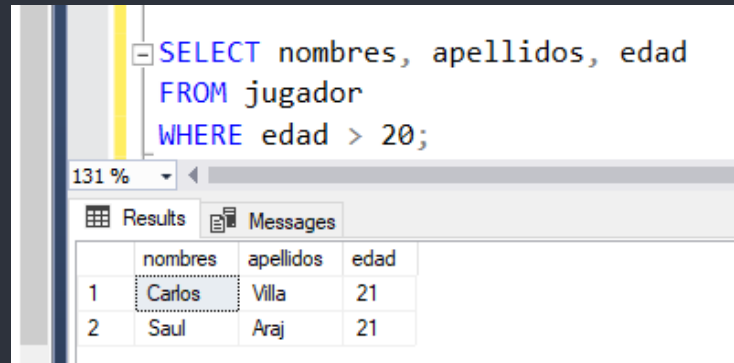
**Ahorro de tiempo:** Simplifica la inserción de registros, ya que no es necesario proporcionar un valor para la columna IDENTITY.

Un ejemplo común de uso de la propiedad IDENTITY es en una tabla de "Clientes", donde se podría usar una columna con IDENTITY para generar automáticamente números de identificación de cliente únicos para cada nuevo cliente registrado en la base de datos. Esto facilita la gestión de los registros y garantiza que cada cliente tenga un identificador único.

</ Para que se utiliza la cláusula WHERE.

La cláusula WHERE en SQL se utiliza para filtrar filas en una consulta SQL, permitiéndote seleccionar solo las filas que cumplan con una condición específica. Esto es útil para recuperar datos específicos de una tabla que cumplan con ciertos criterios.

Aquí tienes un ejemplo en la base de datos "UNIFRANZITOS" que utiliza la cláusula WHERE para filtrar jugadores mayores de 20 años:



The screenshot shows a SQL query editor with the following text:

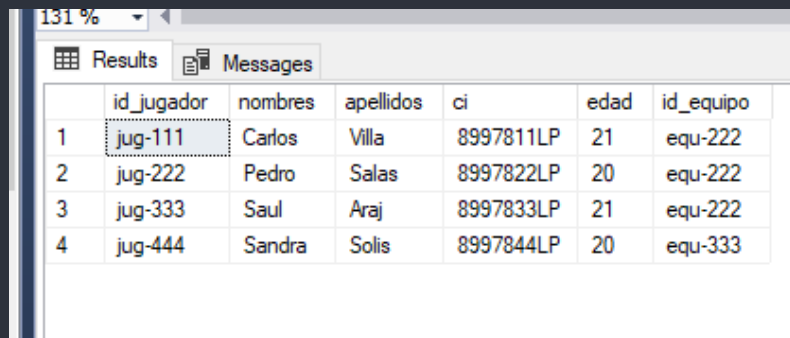
```
SELECT nombres, apellidos, edad  
FROM jugador  
WHERE edad > 20;
```

Below the query, there is a results pane with a zoom level of 131%. It contains two tabs: "Results" and "Messages". The "Results" tab is active, displaying a table with the following data:

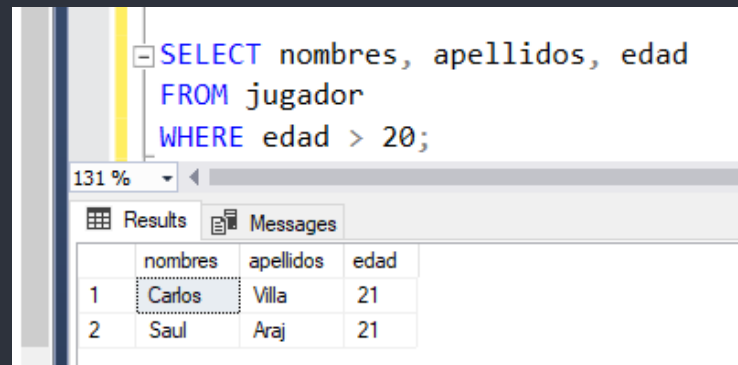
	nombres	apellidos	edad
1	Carlos	Villa	21
2	Saul	Araj	21

</ Para que se utiliza la cláusula WHERE.

Debemos recordar que nuestra tabla quedo de la siguiente manera debido a que anteriormente cambiamos la edad de un jugador [UPDATE] y también eliminamos a un jugador [DELETE], por lo que solo tendríamos a dos jugadores con la edad mayor a 20.



	id_jugador	nombres	apellidos	ci	edad	id_equipo
1	jug-111	Carlos	Villa	8997811LP	21	equ-222
2	jug-222	Pedro	Salas	8997822LP	20	equ-222
3	jug-333	Saul	Araj	8997833LP	21	equ-222
4	jug-444	Sandra	Solis	8997844LP	20	equ-333



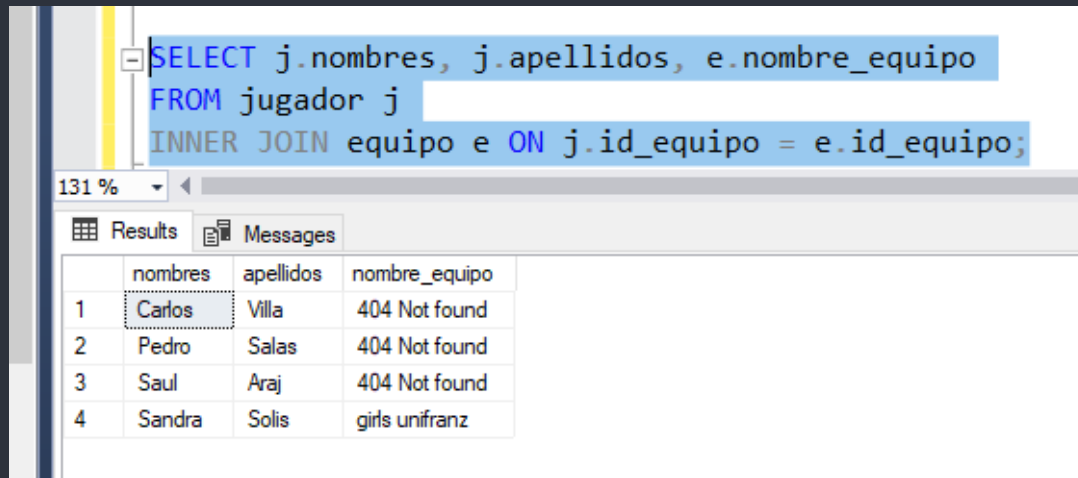
```
SELECT nombres, apellidos, edad
FROM jugador
WHERE edad > 20;
```

	nombres	apellidos	edad
1	Carlos	Villa	21
2	Saul	Araj	21



</ Para que se utiliza la instrucción INNER JOIN.

La instrucción INNER JOIN se utiliza en SQL para combinar filas de dos o más tablas basándose en una condición de igualdad entre las columnas de esas tablas. El resultado de esta combinación es una única tabla que contiene todas las filas de las tablas de origen que cumplen con la condición especificada en la cláusula INNER JOIN.



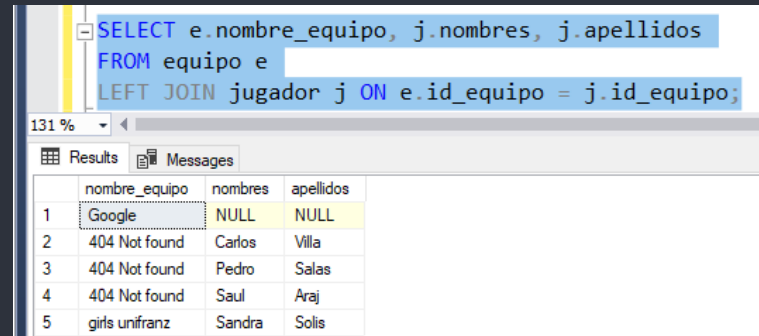
```
SELECT j.nombres, j.apellidos, e.nombre_equipo
FROM jugador j
INNER JOIN equipo e ON j.id_equipo = e.id_equipo;
```

	nombres	apellidos	nombre_equipo
1	Carlos	Villa	404 Not found
2	Pedro	Salas	404 Not found
3	Saul	Araj	404 Not found
4	Sandra	Solis	girls unifranz

## </ Para que se utiliza la instrucción e LEFT JOIN

La instrucción LEFT JOIN se utiliza en SQL para combinar filas de dos o más tablas, mostrando todas las filas de la tabla de la izquierda [tabla principal] y las filas coincidentes de la tabla de la derecha. Si no hay correspondencia en la tabla de la derecha, se mostrarán valores nulos en las columnas de esa tabla.

La principal diferencia entre INNER JOIN y LEFT JOIN es que INNER JOIN solo muestra las filas que tienen una correspondencia en ambas tablas, mientras que LEFT JOIN muestra todas las filas de la tabla izquierda y las filas coincidentes de la tabla derecha.



The screenshot shows a SQL query editor with the following query:

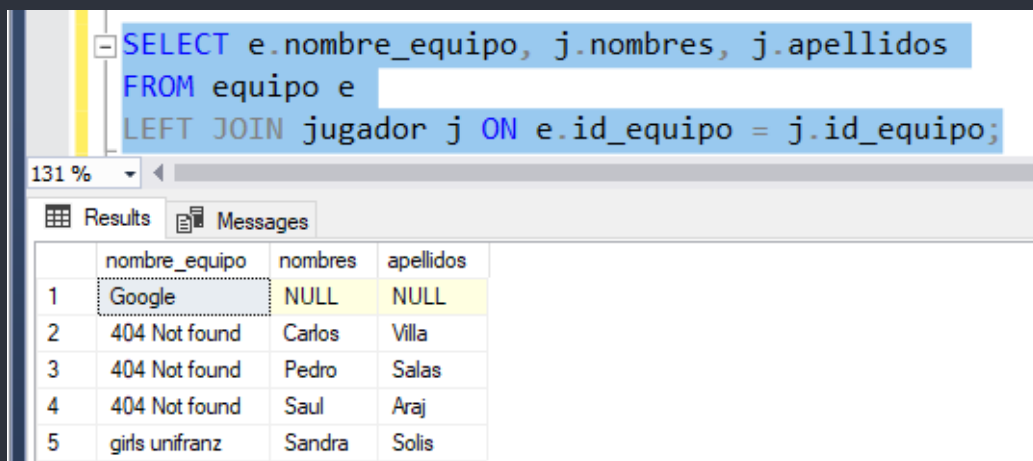
```
SELECT e.nombre_equipo, j.nombres, j.apellidos  
FROM equipo e  
LEFT JOIN jugador j ON e.id_equipo = j.id_equipo;
```

Below the query, the 'Results' tab is active, displaying a table with 5 rows and 3 columns: nombre\_equipo, nombres, and apellidos. The first row shows 'Google' for nombre\_equipo, and NULL for both nombres and apellidos. The subsequent rows show '404 Not found' for nombre\_equipo and player names for nombres and apellidos.

	nombre_equipo	nombres	apellidos
1	Google	NULL	NULL
2	404 Not found	Carlos	Villa
3	404 Not found	Pedro	Salas
4	404 Not found	Saul	Araj
5	girls unifranz	Sandra	Solis

</ Para que se utiliza la instrucción e LEFT JOIN

En esta consulta, se utiliza un LEFT JOIN entre la tabla "equipo" [e] y la tabla "jugador" [j] en función de la igualdad entre e.id\_equipo y j.id\_equipo. El resultado mostrará una lista de equipos, y si hay jugadores asociados a un equipo, también se mostrarán los nombres y apellidos de esos jugadores. Si un equipo no tiene jugadores, las columnas de jugadores mostrarán valores nulos.



The screenshot shows a SQL query editor with the following query:

```
SELECT e.nombre_equipo, j.nombres, j.apellidos  
FROM equipo e  
LEFT JOIN jugador j ON e.id_equipo = j.id_equipo;
```

Below the query, the results are displayed in a table with 4 columns: an index, nombre\_equipo, nombres, and apellidos. The results show 5 rows. The first row has 'Google' in nombre\_equipo and NULL in the other two columns. The next three rows have '404 Not found' in nombre\_equipo and player names in the other two columns. The last row has 'girls unifranz' in nombre\_equipo and player names in the other two columns.

	nombre_equipo	nombres	apellidos
1	Google	NULL	NULL
2	404 Not found	Carlos	Villa
3	404 Not found	Pedro	Salas
4	404 Not found	Saul	Araj
5	girls unifranz	Sandra	Solis

## </ Para que se utiliza la instrucción e RIGHT JOIN

La instrucción RIGHT JOIN se utiliza en SQL para combinar filas de dos o más tablas, mostrando todas las filas de la tabla de la derecha [tabla principal] y las filas coincidentes de la tabla de la izquierda. Si no hay correspondencia en la tabla de la izquierda, se mostrarán valores nulos en las columnas de esa tabla.

La principal diferencia entre INNER JOIN y RIGHT JOIN es que INNER JOIN solo muestra las filas que tienen una correspondencia en ambas tablas, mientras que RIGHT JOIN muestra todas las filas de la tabla de la derecha y las filas coincidentes de la tabla izquierda.

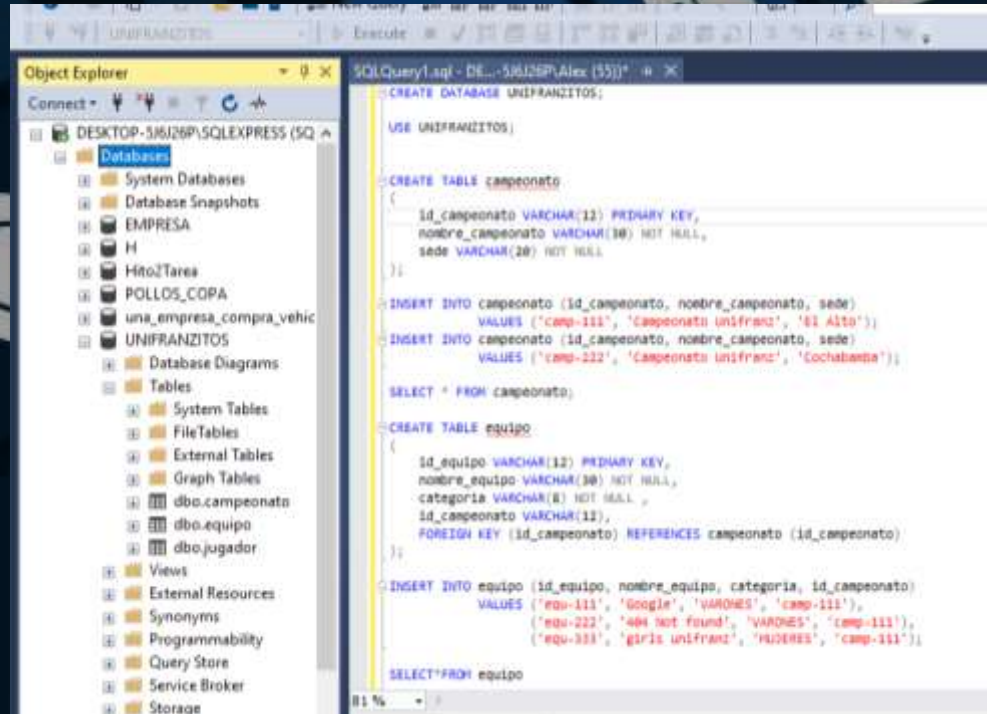
</ Para que se utiliza la instrucción e RIGHT JOIN

En esta consulta, se utiliza un RIGHT JOIN entre la tabla "jugador" (j) y la tabla "equipo" (e) en función de la igualdad entre j.id\_equipo y e.id\_equipo. El resultado mostrará una lista de jugadores, y si están asociados a un equipo, también se mostrará el nombre del equipo correspondiente. Si un jugador no está asociado a ningún equipo, las columnas de equipo mostrarán valores nulos.

```
SELECT j.nombres, j.apellidos, e.nombre_equipo
FROM jugador j
RIGHT JOIN equipo e ON j.id_equipo = e.id_equipo;
```

	nombres	apellidos	nombre_equipo
1	NULL	NULL	Google
2	Carlos	Villa	404 Not found
3	Pedro	Salas	404 Not found
4	Saul	Araj	404 Not found
5	Sandra	Solis	girls unifranz

# </ III.- Manejo de consultas



The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' pane shows the 'Databases' folder expanded, with 'UNIFRANZITOS' selected. Below it, the 'Tables' folder is also expanded, showing 'dbo.campeonato', 'dbo.equipo', and 'dbo.jugador'. The right pane shows a SQL query script named 'SQLQuery1.sql' with the following content:

```
CREATE DATABASE UNIFRANZITOS;

USE UNIFRANZITOS;

CREATE TABLE campeonato
(
    Id_campeonato VARCHAR(11) PRIMARY KEY,
    nombre_campeonato VARCHAR(30) NOT NULL,
    sede VARCHAR(20) NOT NULL
);

INSERT INTO campeonato (Id_campeonato, nombre_campeonato, sede)
VALUES ('camp-111', 'Campeonato Unifranz', 'El Alto');

INSERT INTO campeonato (Id_campeonato, nombre_campeonato, sede)
VALUES ('camp-222', 'Campeonato Unifranz', 'Cochabamba');

SELECT * FROM campeonato;

CREATE TABLE equipo
(
    Id_equipo VARCHAR(12) PRIMARY KEY,
    nombre_equipo VARCHAR(30) NOT NULL,
    categoria VARCHAR(8) NOT NULL,
    Id_campeonato VARCHAR(11),
    FOREIGN KEY (Id_campeonato) REFERENCES campeonato (Id_campeonato)
);

INSERT INTO equipo (Id_equipo, nombre_equipo, categoria, Id_campeonato)
VALUES ('equ-111', 'Google', 'VARONES', 'camp-111'),
('equ-222', '404 Not Found', 'VARONES', 'camp-111'),
('equ-333', 'girls unifranz', 'MUJERES', 'camp-111');

SELECT * FROM equipo
```

## </ Mostrar que jugadores que son del equipo equ-222

Para mostrar los jugadores que son del equipo con el identificador "equ-222" en la base de datos "UNIFRANZITOS", puedes utilizar la cláusula INNER JOIN para combinar las tablas "jugador" y "equipo" y aplicar una condición que filtre los jugadores del equipo específico:

```
SELECT j.nombres, j.apellidos
FROM jugador j
INNER JOIN equipo e ON j.id_equipo = e.id_equipo
WHERE e.id_equipo = 'equ-222';
```

131 %

Results Messages

	nombres	apellidos
1	Carlos	Villa
2	Pedro	Salas
3	Saul	Araj

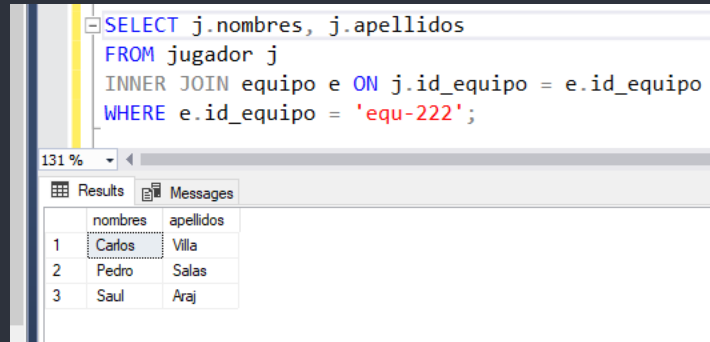
## </ Mostrar que jugadores que son del equipo equ-222

En esta consulta:

Utilizamos un INNER JOIN para combinar las tablas "jugador" [j] y "equipo" [e] basándonos en la igualdad entre j.id\_equipo y e.id\_equipo.

La cláusula WHERE se utiliza para aplicar una condición que filtra los resultados. En este caso, estamos seleccionando solamente las filas donde el identificador del equipo en la tabla "equipo" sea igual a 'equ-222'.

Como resultado, obtendrás una lista de los nombres y apellidos de los jugadores que son miembros del equipo 'equ-222' en la base de datos "UNIFRANZITOS".:



The screenshot shows a SQL query editor with the following text:

```
SELECT j.nombres, j.apellidos  
FROM jugador j  
INNER JOIN equipo e ON j.id_equipo = e.id_equipo  
WHERE e.id_equipo = 'equ-222';
```

Below the query, the 'Results' tab is active, displaying a table with 3 rows and 2 columns: 'nombres' and 'apellidos'. The first row is highlighted with a dashed border.

	nombres	apellidos
1	Carlos	Villa
2	Pedro	Salas
3	Saul	Araj



## </ Mostrar jugadores[nombres, apellidos] que juegan en la sede de El Alto

Para mostrar los jugadores [nombres y apellidos] que juegan en la sede de "El Alto" en la base de datos "UNIFRANZITOS", puedes utilizar una consulta que incluya una combinación de las tablas "jugador", "equipo" y "campeonato", y aplicar una condición para filtrar los jugadores en función de la sede del campeonato. Aquí está la consulta:

```
SELECT j.nombres, j.apellidos  
FROM jugador j  
INNER JOIN equipo e ON j.id_equipo = e.id_equipo  
INNER JOIN campeonato c ON e.id_campeonato = c.id_campeonato  
WHERE c.sede = 'El Alto';
```

131 %

Results

Messages

	nombres	apellidos
1	Carlos	Villa
2	Pedro	Salas
3	Saul	Araj
4	Sandra	Solis

</ Mostrar que jugadores[nombres, apellidos] que juegan en la sede de El Alto

En esta consulta:

Utilizamos un INNER JOIN para combinar las tablas "jugador" [j] y "equipo" [e] basándonos en la igualdad entre j.id\_equipo y e.id\_equipo.

Luego, utilizamos otro INNER JOIN para combinar la tabla "equipo" con la tabla "campeonato" [c] en función de la igualdad entre e.id\_campeonato y c.id\_campeonato.

La cláusula WHERE se utiliza para aplicar una condición que filtra los resultados. En este caso, estamos seleccionando solamente las filas donde la sede del campeonato en la tabla "campeonato" sea igual a 'El Alto'.

```
SELECT j.nombres, j.apellidos
FROM jugador j
INNER JOIN equipo e ON j.id_equipo = e.id_equipo
INNER JOIN campeonato c ON e.id_campeonato = c.id_campeonato
WHERE c.sede = 'El Alto';
```

	nombres	apellidos
1	Carlos	Villa
2	Pedro	Salas
3	Saul	Araj
4	Sandra	Solis

RECORDEMOS QUE NOS FALTA  
UN JUGADOR QUE LO  
ELIMINAMOS ANTERIORMENTE.

</ Mostrar aquellos jugadores mayores o igual a 21 años que sean de la categoría VARONES

Para mostrar los jugadores que tienen 21 años o más y pertenecen a la categoría VARONES en la base de datos "UNIFRANZITOS", puedes utilizar la siguiente consulta SQL:

```
SELECT nombres, apellidos, edad
FROM jugador
INNER JOIN equipo ON jugador.id_equipo = equipo.id_equipo
WHERE edad >= 21 AND equipo.categoria = 'VARONES';
```

131 %

Results Messages

	nombres	apellidos	edad
1	Carlos	Villa	21
2	Saul	Araj	21

RECORDEMOS QUE NOS FALTA  
UN JUGADOR QUE LO  
ELIMINAMOS ANTERIORMENTE Y  
TAMBIEN QUE UN JUGADOR  
CAMBIO SU EDAD DE 19 A 21.

</ Mostrar aquellos jugadores mayores o igual a 21 años que sean de la categoría VARONES

En esta consulta:

Utilizamos un INNER JOIN para combinar las tablas "jugador" y "equipo" en función de la igualdad entre jugador.id\_equipo y equipo.id\_equipo.

Luego, aplicamos una condición en la cláusula WHERE para seleccionar las filas en las que la edad [edad] del jugador sea mayor o igual a 21 años y la categoría del equipo [equipo.categoria] sea igual a 'VARONES'.

```
SELECT nombres, apellidos, edad
FROM jugador
INNER JOIN equipo ON jugador.id_equipo = equipo.id_equipo
WHERE edad >= 21 AND equipo.categoria = 'VARONES';
```

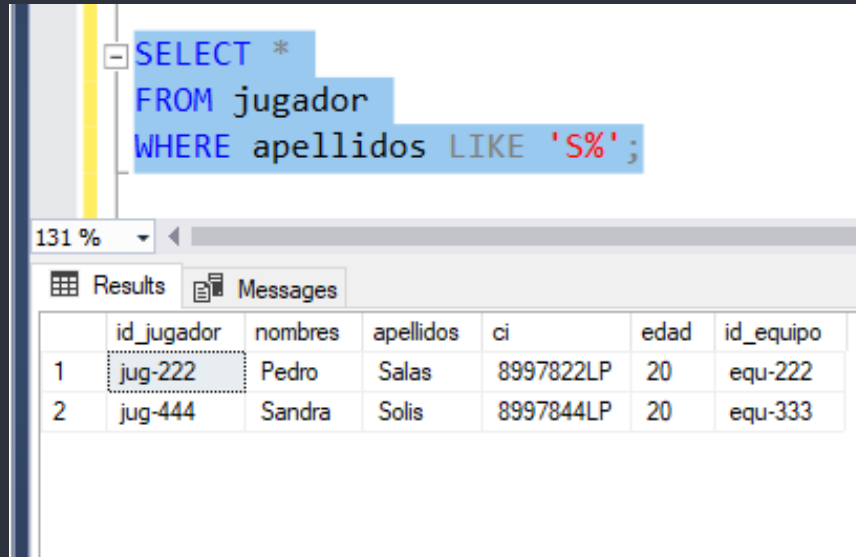
RECORDEMOS QUE NOS FALTA  
UN JUGADOR QUE LO  
ELIMINAMOS ANTERIORMENTE Y  
TAMBIEN QUE UN JUGADOR  
CAMBIO SU EDAD DE 19 A 21.

131 %

	nombres	apellidos	edad
1	Carlos	Villa	21
2	Saul	Araj	21

</ Mostrar a todos los estudiantes en donde su apellido empiece con la letra S.

Para mostrar a todos los estudiantes cuyos apellidos comienzan con la letra "S" en la base de datos "UNIFRANZITOS", puedes utilizar la instrucción LIKE en una consulta SQL



```
SELECT *  
FROM jugador  
WHERE apellidos LIKE 'S%';
```

	id_jugador	nombres	apellidos	ci	edad	id_equipo
1	jug-222	Pedro	Salas	8997822LP	20	equ-222
2	jug-444	Sandra	Solis	8997844LP	20	equ-333

RECORDEMOS QUE NOS FALTA  
UN JUGADOR QUE LO  
ELIMINAMOS ANTERIORMENTE Y  
TAMBIEN QUE UN JUGADOR  
CAMBIO SU EDAD DE 19 A 21.

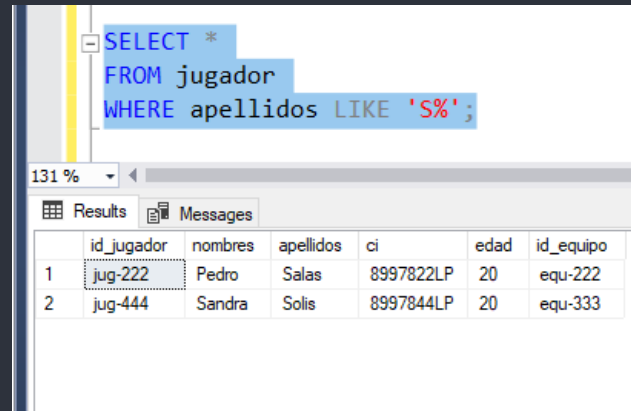
</ Mostrar a todos los estudiantes en donde su apellido empiece con la letra S.

En esta consulta:

Utilizamos la cláusula WHERE para aplicar una condición de filtrado.

En la condición, utilizamos LIKE seguido de 'S%' para indicar que estamos buscando todos los apellidos que comienzan con la letra 'S'. El símbolo '%' se utiliza como comodín, lo que significa que puede haber cualquier número de caracteres después de la 'S'.

RECORDEMOS QUE NOS FALTA  
UN JUGADOR QUE LO  
ELIMINAMOS ANTERIORMENTE Y  
TAMBIEN QUE UN JUGADOR  
CAMBIO SU EDAD DE 19 A 21.



The screenshot shows a SQL query editor with the following query:

```
SELECT *  
FROM jugador  
WHERE apellidos LIKE 'S%';
```

Below the query editor, the results are displayed in a table. The table has columns: id\_jugador, nombres, apellidos, ci, edad, and id\_equipo. The results show two rows:

	id_jugador	nombres	apellidos	ci	edad	id_equipo
1	jug-222	Pedro	Salas	8997822LP	20	equ-222
2	jug-444	Sandra	Solis	8997844LP	20	equ-333

</ Mostrar que equipos forman parte del campeonato camp-111 y además sean de la categoría MUJERES.

Para mostrar qué equipos forman parte del campeonato "camp-111" y además son de la categoría "MUJERES" en la base de datos "UNIFRANZITOS", puedes utilizar la siguiente consulta SQL:

```
SELECT e.nombre_equipo, e.categoria
FROM equipo e
INNER JOIN campeonato c ON e.id_campeonato = c.id_campeonato
WHERE c.id_campeonato = 'camp-111' AND e.categoria = 'MUJERES';
```

31 %

Results Messages

	nombre_equipo	categoria
1	girls unifranz	MUJERES

</ Mostrar a todos los estudiantes en donde su apellido empiece con la letra S.

En esta consulta:

Utilizamos un INNER JOIN para combinar las tablas "equipo" [e] y "campeonato" [c] en función de la igualdad entre e.id\_campeonato y c.id\_campeonato.

Luego, aplicamos una condición en la cláusula WHERE para seleccionar las filas en las que el identificador del campeonato en la tabla "campeonato" sea igual a 'camp-111' y la categoría del equipo en la tabla "equipo" sea igual a 'MUJERES'.

SOLO EXISTE UN EQUIPO DE CATEGORIA MUJERES



</ Mostrar el nombre del equipo del jugador con id\_jugador igual a jug-333

Para mostrar el nombre del equipo al que pertenece un jugador con id\_jugador igual a 'jug-333' en nuestra base de datos "UNIFRANZITOS", puedes utilizar la siguiente consulta SQL:

```
SELECT e.nombre_equipo
FROM equipo e
INNER JOIN jugador j ON e.id_equipo = j.id_equipo
WHERE j.id_jugador = 'jug-333';
```

131 %

Results Messages

	nombre_equipo
1	404 Not found

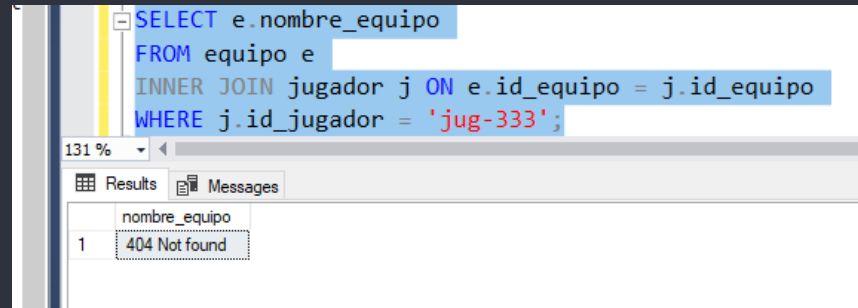
## </ Mostrar el nombre del equipo del jugador con id\_jugador igual a jug-333

En esta consulta:

Utilizamos un INNER JOIN para combinar las tablas "equipo" [e] y "jugador" [j] en función de la igualdad entre e.id\_equipo y j.id\_equipo.

Luego, aplicamos una condición en la cláusula WHERE para seleccionar las filas en las que el identificador del jugador en la tabla "jugador" sea igual a 'jug-333'.

Como resultado, obtendrás el nombre del equipo al que pertenece el jugador con id\_jugador igual a 'jug-333' en la base de datos "UNIFRANZITOS".



```
SELECT e.nombre_equipo
FROM equipo e
INNER JOIN jugador j ON e.id_equipo = j.id_equipo
WHERE j.id_jugador = 'jug-333';
```

131 %

Results Messages

	nombre_equipo
1	404 Not found

</ Mostrar el nombre del campeonato del jugador con id\_jugador igual a jug-333

Para mostrar el nombre del campeonato al que pertenece un jugador con id\_jugador igual a 'jug-333' en la base de datos "UNIFRANZITOS", puedes utilizar la siguiente consulta SQL:

```
SELECT c.nombre_campeonato
FROM campeonato c
INNER JOIN equipo e ON c.id_campeonato = e.id_campeonato
INNER JOIN jugador j ON e.id_equipo = j.id_equipo
WHERE j.id_jugador = 'jug-333';
```

131 %

Results Messages

	nombre_campeonato
1	Campeonato Unifranz

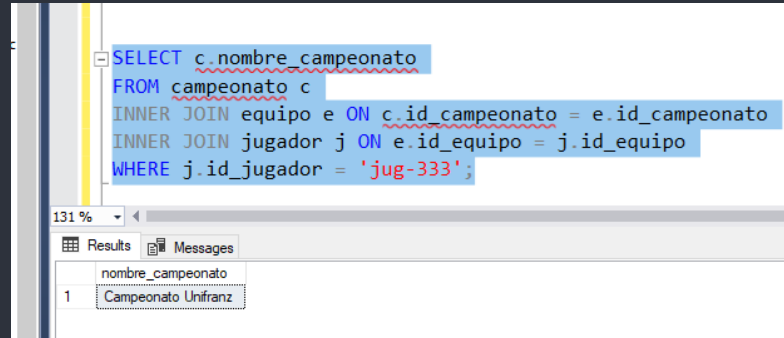
</ Mostrar el nombre del campeonato del jugador con id\_jugador igual a jug-333

En esta consulta:

Utilizamos múltiples INNER JOIN para combinar las tablas "campeonato" [c], "equipo" [e] y "jugador" [j] en función de las igualdades entre c.id\_campeonato, e.id\_campeonato, e.id\_equipo y j.id\_equipo.

Luego, aplicamos una condición en la cláusula WHERE para seleccionar las filas en las que el identificador del jugador en la tabla "jugador" sea igual a 'jug-333'.

Como resultado, obtendrás el nombre del campeonato al que pertenece el jugador con id\_jugador igual a 'jug-333' en la base de datos "UNIFRANZITOS".



```
SELECT c.nombre_campeonato
FROM campeonato c
INNER JOIN equipo e ON c.id_campeonato = e.id_campeonato
INNER JOIN jugador j ON e.id_equipo = j.id_equipo
WHERE j.id_jugador = 'jug-333';
```

nombre_campeonato
1   Campeonato Unifranz

</Crear una consulta SQL que maneje las 3 tablas de la base de datos.

Para crear una consulta SQL que involucre las tres tablas de la base de datos "UNIFRANZITOS" [campeonato, equipo y jugador], puedes utilizar una consulta que combine la información de estas tablas en función de las relaciones establecidas. Aquí tienes un ejemplo de consulta que muestra información de campeonatos, equipos y jugadores:

```
SELECT c.id_campeonato, c.nombre_campeonato, c.sede, e.id_equipo, e.nombre_equipo, e.categoria, j.id_jugador, j.nombres, j.apellidos, j.ci, j.edad  
FROM campeonato c  
INNER JOIN equipo e ON c.id_campeonato = e.id_campeonato  
INNER JOIN jugador j ON e.id_equipo = j.id_equipo;
```

98 %

Results Messages

	id_campeonato	nombre_campeonato	sede	id_equipo	nombre_equipo	categoria	id_jugador	nombres	apellidos	ci	edad
1	camp-111	Campeonato Unifranz	El Alto	equ-222	404 Not found	VARONES	jug-111	Carlos	Villa	8997811LP	21
2	camp-111	Campeonato Unifranz	El Alto	equ-222	404 Not found	VARONES	jug-222	Pedro	Salas	8997822LP	20
3	camp-111	Campeonato Unifranz	El Alto	equ-222	404 Not found	VARONES	jug-333	Saul	Araj	8997833LP	21
4	camp-111	Campeonato Unifranz	El Alto	equ-333	girls unifranz	MUJERES	jug-444	Sandra	Solis	8997844LP	20

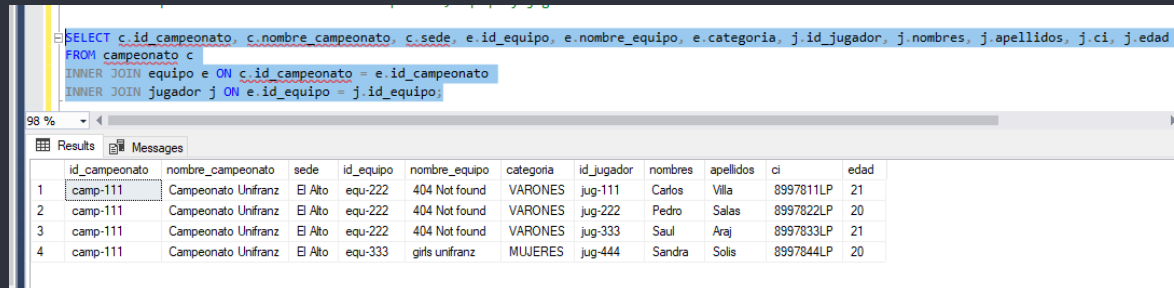
## </Crear una consulta SQL que maneje las 3 tablas de la base de datos.

En esta consulta:

Utilizamos múltiples INNER JOIN para combinar las tablas "campeonato" [c], "equipo" [e] y "jugador" [j] en función de las igualdades entre los identificadores de campeonato y equipo, así como entre los identificadores de equipo y jugador.

Seleccionamos diversas columnas de las tres tablas, como el identificador y nombre del campeonato, el identificador y nombre del equipo, y la información del jugador, como nombres, apellidos, cédula de identidad [ci], edad, etc.

Esta consulta te proporcionará una vista combinada de la información de campeonatos, equipos y jugadores en la base de datos "UNIFRANZITOS". Puedes personalizar la consulta para incluir las columnas y criterios que sean relevantes para tu análisis específico.



The screenshot shows a SQL query editor with the following query:

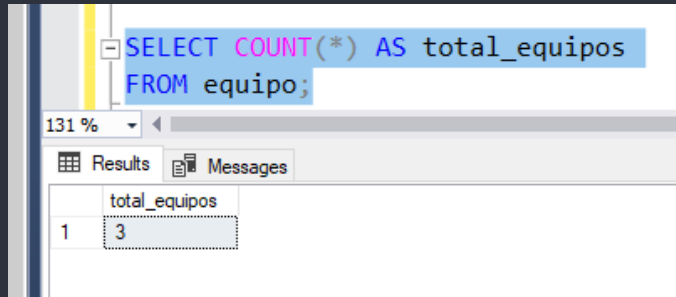
```
SELECT c.id_campeonato, c.nombre_campeonato, c.sede, e.id_equipo, e.nombre_equipo, e.categoria, j.id_jugador, j.nombres, j.apellidos, j.ci, j.edad
FROM campeonato c
INNER JOIN equipo e ON c.id_campeonato = e.id_campeonato
INNER JOIN jugador j ON e.id_equipo = j.id_equipo;
```

Below the query, the results are displayed in a table with 11 columns: id\_campeonato, nombre\_campeonato, sede, id\_equipo, nombre\_equipo, categoria, id\_jugador, nombres, apellidos, ci, and edad. The results show 4 rows of data, all from the 'camp-111' tournament.

	id_campeonato	nombre_campeonato	sede	id_equipo	nombre_equipo	categoria	id_jugador	nombres	apellidos	ci	edad
1	camp-111	Campeonato Unifranz	El Alto	equ-222	404 Not found	VARONES	jug-111	Carlos	Villa	8997811LP	21
2	camp-111	Campeonato Unifranz	El Alto	equ-222	404 Not found	VARONES	jug-222	Pedro	Salas	8997822LP	20
3	camp-111	Campeonato Unifranz	El Alto	equ-222	404 Not found	VARONES	jug-333	Saul	Araj	8997833LP	21
4	camp-111	Campeonato Unifranz	El Alto	equ-333	girls unifranz	MUJERES	jug-444	Sandra	Solis	8997844LP	20

## </ ¿Qué estrategia utilizaría para determinar cuántos equipos inscritos hay?

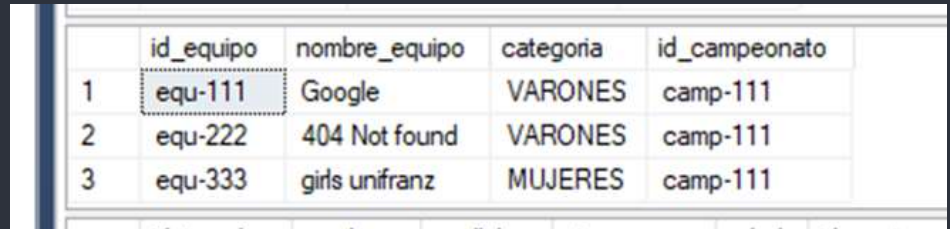
Para determinar cuántos equipos están inscritos en la base de datos "UNIFRANZITOS", puedes utilizar una consulta SQL que cuente el número de filas en la tabla "equipo". La estrategia más sencilla sería la siguiente:



The screenshot shows a SQL query editor with the following query: `SELECT COUNT(*) AS total_equipos FROM equipo;`. Below the query, the 'Results' tab is active, displaying a single row with the value '3' under the column 'total\_equipos'.

	total_equipos
1	3

Como vemos, si hay y son tres equipos inscritos



The screenshot shows a table with the following data:

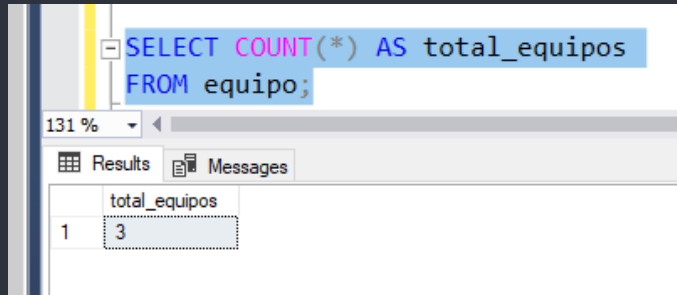
	id_equipo	nombre_equipo	categoria	id_campeonato
1	equ-111	Google	VARONES	camp-111
2	equ-222	404 Not found	VARONES	camp-111
3	equ-333	girls unifranz	MUJERES	camp-111

## </ ¿Qué estrategia utilizaría para determinar cuántos equipos inscritos hay?

En esta consulta, estamos utilizando la función COUNT(\*), que cuenta todas las filas en la tabla "equipo". El resultado de esta consulta será el número total de equipos inscritos en la base de datos "UNIFRANZITOS".

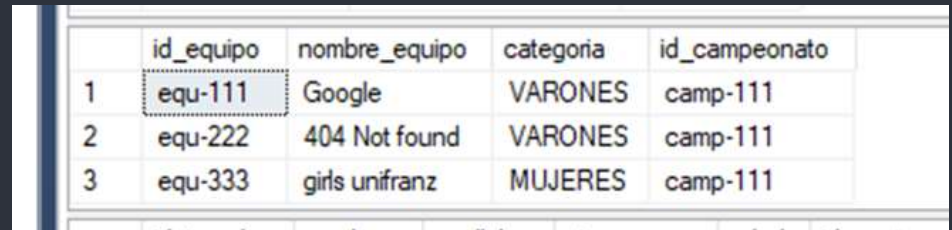
Al usar AS total\_equipos, le estamos dando un alias al resultado para que sea más legible. El número total de equipos inscritos se mostrará bajo el nombre "total\_equipos" en el resultado de la consulta.

Esta estrategia te proporcionará de manera rápida y sencilla el conteo de equipos inscritos en la base de datos. Puedes ejecutar esta consulta directamente en tu sistema de gestión de base de datos o en la interfaz que estés utilizando para interactuar con la base de datos.



The screenshot shows a SQL query editor with the following query: `SELECT COUNT(*) AS total_equipos FROM equipo;`. Below the query, the results are displayed in a table with one column, `total_equipos`, and one row with the value `3`.

	total_equipos
1	3



The screenshot shows a table with five columns: `id_equipo`, `nombre_equipo`, `categoria`, and `id_campeonato`. The table contains three rows of data.

	id_equipo	nombre_equipo	categoria	id_campeonato
1	equ-111	Google	VARONES	camp-111
2	equ-222	404 Not found	VARONES	camp-111
3	equ-333	girls unifranz	MUJERES	camp-111

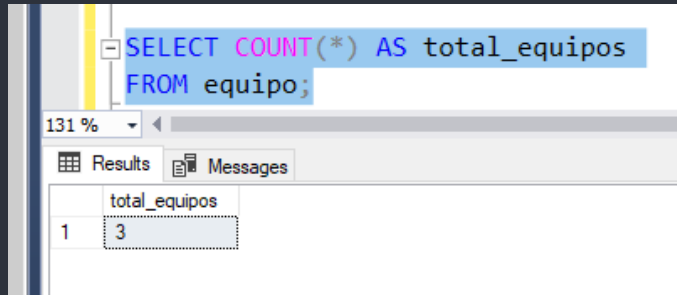


## </ ¿Qué estrategia utilizaría para determinar cuántos equipos inscritos hay?

En esta consulta, estamos utilizando la función COUNT(\*), que cuenta todas las filas en la tabla "equipo". El resultado de esta consulta será el número total de equipos inscritos en la base de datos "UNIFRANZITOS".

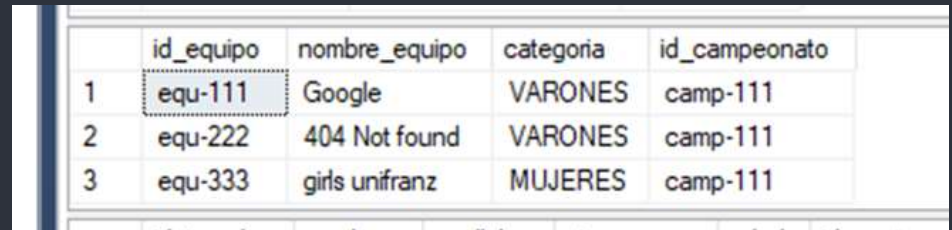
Al usar AS total\_equipos, le estamos dando un alias al resultado para que sea más legible. El número total de equipos inscritos se mostrará bajo el nombre "total\_equipos" en el resultado de la consulta.

Esta estrategia te proporcionará de manera rápida y sencilla el conteo de equipos inscritos en la base de datos. Puedes ejecutar esta consulta directamente en tu sistema de gestión de base de datos o en la interfaz que estés utilizando para interactuar con la base de datos.



The screenshot shows a SQL query editor with the following query: `SELECT COUNT(*) AS total_equipos FROM equipo;`. Below the query, the results are displayed in a table with one column, `total_equipos`, and one row with the value `3`.

	total_equipos
1	3



The screenshot shows a table with five columns: `id_equipo`, `nombre_equipo`, `categoria`, and `id_campeonato`. The table contains three rows of data.

	id_equipo	nombre_equipo	categoria	id_campeonato
1	equ-111	Google	VARONES	camp-111
2	equ-222	404 Not found	VARONES	camp-111
3	equ-333	girls unifranz	MUJERES	camp-111

</ ¿Qué estrategia utilizaría para determinar cuántos jugadores pertenecen a la categoría VARONES o Categoría MUJERES.

Para determinar cuántos jugadores pertenecen a la categoría "VARONES" o "MUJERES" utilizando la función COUNT, puedes hacerlo utilizando una consulta que agrupe los jugadores por categoría y luego cuente el número de jugadores en cada categoría. Aquí tienes la estrategia:

```
SELECT e.categoria, COUNT(*) AS total_jugadores
FROM jugador j
INNER JOIN equipo e ON j.id_equipo = e.id_equipo
WHERE e.categoria IN ('VARONES', 'MUJERES')
GROUP BY e.categoria;
```

131 %

Results Messages

	categoria	total_jugadores
1	MUJERES	1
2	VARONES	3

RECORDEMOS QUE NOS FALTA  
UN JUGADOR QUE LO  
ELIMINAMOS ANTERIORMENTE Y  
TAMBIEN QUE UN JUGADOR  
CAMBIO SU EDAD DE 19 A 21.

</ ¿Qué estrategia utilizaría para determinar cuántos jugadores pertenecen a la categoría VARONES o Categoría MUJERES.

En esta consulta:

Utilizamos un INNER JOIN para combinar las tablas "jugador" [j] y "equipo" [e] en función de la igualdad entre j.id\_equipo y e.id\_equipo.

La cláusula WHERE se utiliza para filtrar las filas donde la categoría del equipo sea 'VARONES' o 'MUJERES'.

La función COUNT[\*] se utiliza para contar el número de jugadores en cada categoría.

La cláusula GROUP BY se utiliza para agrupar los resultados por la columna "categoria" de la tabla "equipo".

El resultado de esta consulta mostrará el número de jugadores en las categorías "VARONES" y "MUJERES". La columna "categoria" mostrará la categoría correspondiente y la columna "total\_jugadores" mostrará el recuento de jugadores en cada categoría.

Esta estrategia utiliza la función COUNT para calcular el número de jugadores en cada categoría y te proporcionará una vista desglosada de los jugadores por categoría.

</ GRACIAS