Consigna

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

UNIFRANZITOS	
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**

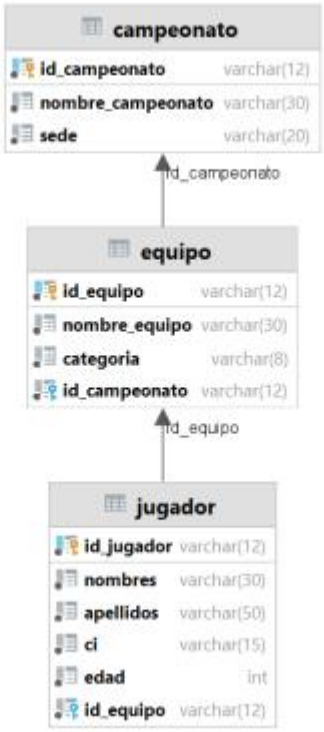
```
Create Database Hito_3
Use Hito_3
Create table Campeonato
```

```
(
  Id_campeonato Varchar(12) Primary key not null,
  Nombre_Campeonato Varchar(30) not null,
  Sede Varchar(20) not null
);

Create table Equipos
 (
   id_equipo Varchar(12) PRIMARY KEY NOT NULL,
   nombre_equipo Varchar (30) NOT NULL,
   categoria Varchar (7) NOT NULL CHECK (categoria in ('varones', 'mujeres')),
   Id_campeonato varchar(12)
         FOREIGN KEY (Id_campeonato) REFERENCES Campeonato(Id_campeonato)
);

Create table Jugador
(
   id_jugador Varchar (12) PRIMARY KEY NOT NULL,
   nombres Varchar (30) NOT NULL,
   apellidos Varchar (50) NOT NULL,
   ci Varchar (15) NOT NULL,
   edad integer NOT NULL,
   id_equipo Varchar (12)
         FOREIGN KEY (id_equipo) REFERENCES Equipos(id_equipo)
);
```

1. Diseño de base de datos.
1.1. Dado el detalle explicado en la parte inicial de este documento debería generar una base de datos similar al
siguiente.



1.2. Los registros de cada tabla deberían quedar de la siguiente forma

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

Insert into Campeonato(Id_campeonato, Nombre_Campeonato, Sede)
values ('camp-111', 'Campeonato Unifranz', 'El Alto'),
('camp-222', 'Campeonato Unifranz', 'Cochabamba')

```
Insert into Equipos (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')

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')
```

2. Manejo de conceptos

2.1. Adjuntar el diagrama E-R GENERADO por su editor (DATAGRIP o SQL SERVER MANAGEMENTS STUDIO)

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

- DDL (Data Definition Language): Se utiliza para definir estructuras de base de datos, como crear, modificar o eliminar tablas. Ejemplo de DDL en SQL Server para crear una tabla en la base de datos UNIFRANZITOS:

```
CREATE TABLE Estudiantes (
    EstudianteID INTEGER PRIMARY KEY,
    Nombre VARCHAR(50),
    Edad INTEGER
);
```

- DML (Data Manipulation Language): Se utiliza para manipular datos en las tablas, como insertar, actualizar o eliminar registros. Ejemplo de DML en SQL Server para insertar datos en la tabla Estudiantes:

```
INSERT INTO Estudiantes (EstudianteID, Nombre, Edad) VALUES (1, 'Juan', 20)
```

2.3 . Que significa PRIMARY KEY y FOREIGN KEY

- PRIMARY KEY: La PRIMARY KEY se utiliza para identificar de manera única cada fila en una tabla Ejemplo:

```
CREATE TABLE Estudiantes (
    EstudianteID INTEGER PRIMARY KEY,
```

```
    Nombre VARCHAR(50),
    Edad INTEGER
);
```

- FOREIGN KEY: La FOREIGN KEY se utiliza para establecer relaciones entre tablas y garantizar que los valores coincidan y mantengan la integridad de los datos en una base de datos relacional. Ejemplo:

```
CREATE TABLE Pedidos
(
    PedidoID INTEGER PRIMARY KEY,
    EstudianteID INTEGER,
    FOREIGN KEY (EstudianteID) REFERENCES Estudiantes(EstudianteID)
);
```

2.4 Defina que es una TABLA y el uso de IDENTITY.

- TABLA: En una base de datos relacional, una tabla es una estructura que organiza los datos en filas y columnas. Cada columna tiene un nombre único y un tipo de datos específico.

- IDENTITY: En SQL Server, `IDENTITY` se utiliza para generar valores de clave primaria únicos automáticamente. Por ejemplo:

```
CREATE TABLE Empleados
(
  ID INT PRIMARY KEY,
  Nombre VARCHAR(50),
  Edad INT,
  Departamento VARCHAR(50)
);
```

```
CREATE TABLE Empleados (
  ID INT PRIMARY KEY IDENTITY(1,1),
  Nombre VARCHAR(50),
  Edad INT,
  Departamento VARCHAR(50)
);
```

2.5 Para que se utiliza la cláusula WHERE.

La cláusula `WHERE` se utiliza en las instrucciones SQL para filtrar las filas que cumplen con una condición específica. Por ejemplo:

```
SELECT * FROM Jugador WHERE edad > 20
```

2.6 Para que se utiliza la instrucción INNER JOIN.

La instrucción `INNER JOIN` se utiliza para mostra filas de dos o más tablas basándose en una condición de relación entre ellas. Por ejemplo:

```
SELECT Jugador.nombres, edad, Equipos.categoria, nombre_equipo
FROM Equipos
INNER JOIN Jugador ON Jugador.id_equipo = Equipos.id_equipo
```

2.7. Apoyándonos en el concepto de conjuntos muestre los siguiente:

2.7.1. Ejemplo de INNER JOIN

```
CREATE TABLE Empleados (
    ID INT,
    Nombre VARCHAR(50),
    Departamento_ID INT
);

INSERT INTO Empleados (ID, Nombre, Departamento_ID)
VALUES (1, 'Juan', 101),
       (2, 'María', 102),
       (3, 'Carlos', 101);

CREATE TABLE Departamentos (
    Departamento_ID INT,
    Nombre VARCHAR(50)
);

INSERT INTO Departamentos (Departamento_ID, Nombre)
VALUES (101, ‘Ventas’),
       (102, ‘Marketing’),
       (103, ‘Finanzas’);

SELECT Empleados.ID, Empleados.Nombre, Empleados.Departamento_ID,
Departamentos.Nombre AS Nombre_Departamento
FROM Empleados
INNER JOIN Departamentos ON Empleados.Departamento_ID =
Departamentos.Departamento_ID;
```

2.7.2. Adjuntar una imagen de conjuntos y la consulta SQL que refleje el INNER JOIN

2.8. Apoyándonos en el concepto de conjuntos muestre los siguiente:

2.8.1. Ejemplo de LEFT JOIN

```
CREATE TABLE Empleados (
    ID INT,
```

```
    Nombre VARCHAR(50),
    Departamento_ID INT
);

INSERT INTO Empleados (ID, Nombre, Departamento_ID)
VALUES (1, 'Juan', 101),
       (2, 'María', 102),
       (3, 'Carlos', 101),
       (4, 'Laura', NULL); -- Laura no tiene departamento asignado

CREATE TABLE Departamentos (
    Departamento_ID INT,
    Nombre VARCHAR(50)
);

INSERT INTO Departamentos (Departamento_ID, Nombre)
VALUES (101, 'Ventas'),
       (102, 'Marketing'),
       (103, 'Finanzas');

SELECT Empleados.ID, Empleados.Nombre, Empleados.Departamento_ID,
ISNULL(Departamentos.Nombre, 'Sin asignar') AS Nombre_Departamento
FROM Empleados
LEFT JOIN Departamentos ON Empleados.Departamento_ID =
Departamentos.Departamento_ID;
```

2.8.2. Adjuntar una imagen de conjuntos y la consulta SQL que refleje el LEFT JOIN

2.9. Apoyándonos en el concepto de conjuntos muestre los siguiente:

2.9.1. Ejemplo de RIGHT JOIN

```
CREATE TABLE Empleados (
    ID INT,
    Nombre VARCHAR(50),
    Departamento_ID INT
);

INSERT INTO Empleados (ID, Nombre, Departamento_ID)
VALUES (1, 'Juan', 101),
       (2, 'María', 102),
       (3, 'Carlos', 101);

CREATE TABLE Departamentos (
    Departamento_ID INT,
    Nombre VARCHAR(50)
);

INSERT INTO Departamentos (Departamento_ID, Nombre)
VALUES (101, 'Ventas'),
```

```
    (102, ‘Marketing’),
    (103, ‘Finanzas’);

SELECT Departamentos.Departamento_ID, Departamentos.Nombre AS
Nombre_Departamento, Empleados.Nombre AS Nombre_Empleado
FROM Departamentos
RIGHT JOIN Empleados ON Departamentos.Departamento_ID =
Empleados.Departamento_ID;
```

2.9.2. Adjuntar una imagen de conjuntos y la consulta SQL que refleje el RIGHT JOIN

2.10 Crear 3 tablas y crear una consulta SQL que muestra el uso de INNER JOIN.

```
  CREATE TABLE Empleados (
   EmpleadoID INTEGER PRIMARY KEY,
   Nombre VARCHAR(50)
);

CREATE TABLE Departamentos (
   DepartamentoID INTEGER PRIMARY KEY,
   Nombre VARCHAR(50)
);

CREATE TABLE EmpleadosDepartamentos (
   EmpleadoID INTEGER,
   DepartamentoID INTEGER,
   FOREIGN KEY (EmpleadoID) REFERENCES Empleados(EmpleadoID),
   FOREIGN KEY (DepartamentoID) REFERENCES Departamentos(DepartamentoID)
);

INSERT INTO Empleados (EmpleadoID, Nombre) VALUES (789456, 'Juan'), (123456,
'Miguel'), (963451, 'Juan'), (743215, 'Miguel')
INSERT INTO Departamentos (DepartamentoID, Nombre) VALUES (456789, 'Miraflores'),
(789123, 'Mallasa'), (157936, 'Juan'), (852496, 'Miguel')
INSERT INTO EmpleadosDepartamentos (EmpleadoID, DepartamentoID) VALUES
(789456, 789123), (123456, 456789)

SELECT Empleados.Nombre AS NombreEmpleado, Departamentos.Nombre AS
NombreDepartamento
FROM Empleados
INNER JOIN EmpleadosDepartamentos ON Empleados.EmpleadoID =
EmpleadosDepartamentos.EmpleadoID
INNER JOIN Departamentos ON EmpleadosDepartamentos.DepartamentoID =
Departamentos.DepartamentoID;
```

Esta consulta `INNER JOIN` te mostrará los nombres de los empleados y los nombres de los departamentos a los que están asignados en función de las relaciones definidas en las tablas.

3. Manejo de consultas
3.1. Mostrar que jugadores que son del equipo equ-222

```
SELECT nombres, apellidos
FROM Jugador
WHERE id_equipo = 'equ-222';
```

3.2. Mostrar que jugadores(nombres, apellidos) que juegan en la sede de El Alto.

```
SELECT nombres, apellidos
FROM Jugador
INNER JOIN Equipos ON Jugador.id_equipo = Equipos.id_equipo
INNER JOIN Campeonato ON Equipos.Id_campeonato = Campeonato.Id_campeonato
WHERE Campeonato.Sede = 'El Alto';
```

3.3. Mostrar aquellos jugadores mayores o igual a 21 años que sean de la categoría VARONES.

```
SELECT nombres, apellidos
FROM Jugador
WHERE edad >= 21 AND id_equipo IN (SELECT id_equipo FROM Equipos WHERE
categoria = 'varones');
```

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

```
SELECT *
FROM Estudiantes
WHERE LEFT(Apellido, 1) = ‘S’;
```

3.4.1. Podría utilizar la instrucción LIKE

```
SELECT nombres, apellidos
FROM Jugador
WHERE apellidos LIKE 'S%';
```

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

```
SELECT nombre_equipo
FROM Equipos
WHERE Id_campeonato = 'camp-111' AND categoria = 'mujeres';
```

3.6. Mostrar el nombre del equipo del jugador con id_jugador igual a jug-333

```
SELECT Equipos.nombre_equipo
```

```
FROM Jugador
INNER JOIN Equipos ON Jugador.id_equipo = Equipos.id_equipo
WHERE id_jugador = 'jug-333';
```

3.7. Mostrar el nombre del campeonato del jugador con id_jugador igual a jug-333

```
SELECT Campeonato.Nombre_Campeonato
FROM Jugador
INNER JOIN Equipos ON Jugador.id_equipo = Equipos.id_equipo
INNER JOIN Campeonato ON Equipos.Id_campeonato = Campeonato.Id_campeonato
WHERE id_jugador = 'jug-333';
```

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

```
SELECT Jugador.nombres AS NombreJugador, Jugador.apellidos AS ApellidoJugador,
Equipos.nombre_equipo AS NombreEquipo, Campeonato.Nombre_Campeonato AS
NombreCampeonato
FROM Jugador
INNER JOIN Equipos ON Jugador.id_equipo = Equipos.id_equipo
INNER JOIN Campeonato ON Equipos.Id_campeonato = Campeonato.Id_campeonato;
```

3.9. ¿Qué estrategia utilizaría para determinar cuántos equipos inscritos hay?

```
SELECT COUNT(*) AS CantidadEquiposInscritos
FROM Equipos;
```