Comandos SQL

En esta práctica se trataran de cubrir gran parte de los comandos SQL. Se enfoca fundamentalmente en el uso de sentencias que contienen esas palabras reservadas que están predefinidas como funciones internas en la Base de datos. SQL (Structured Query Languaje), lenguaje estructurado de consulta, es un lenguaje que permite interactuar con la base de datos por medio de consultas; en práctica se evalúa la sintaxis de las declaraciones y no se profundizara en los componentes del sistema de gestión de una base de datos.

La práctica consiste en una serie de ejemplos basados en los datos de los archivo .csv contenido en el mismo repositorio; dicho archivo contiene datos de los jugadores de futbol en el videojuego FIFA18 actualizados a septiembre de 2017. Primero se muestra la versión general de cada sentencia y luego un caso específico. La descripción general del dataset está en otro archivo .pdf adjunto en este repositorio.

Esta práctica se llevó a cabo a través de la extensión “SQLite manager” del navegador Firefox; este aplicativo presenta ciertas limitantes y no es recomendable para fines despliegue en producción, pero su simplicidad en la instalación resulta muy útil para fines de aprendizaje y cumple con los propósitos de esta práctica.

Esta base de datos relacional generara sus respuestas en forma de tablas (filas con diferentes columnas); se recomienda tener cuidado con el manejo de los punto y coma“;” de las declaraciones.

1. SELECT *

SELECT * FROM nombre_tabla;

SELECT * FROM *fifa18*;

Selecciona todos los registros de la tabla fifa1. Con "*" se indica que se seleccionan todas las columnas.

2. SELECT

SELECT columna1,columna2,…
FROM tabla;

SELECT *full_name, club, age, league, overall FROM fifa1*;

Se hace la selección de solo algunas columnas en la tabla; las que se especifican después de la cláusula SELECT

3. SELECT DISTINCT :

SELECT DISTINCT columna1,columna2,…
FROM tabla;

SELECT DISTINCT *league*
FROM *fifa1*;

Distinct se utiliza para no tener valores duplicados. Con la consulta de ejemplo se obtienen únicamente las ligas de futbol, sin repetir ninguna de ellas.

4. WHERE

SELECT columna1,columna2,…
FROM tabla;
WHERE condicion;

SELECT *full_name,club,age,league,overall*
FROM *fifa1*
WHERE *nationality='Colombia'*;

Se utiliza para añadir condiciones a las columnas de los registro
Con las consulta de ejemplo se obtienen solo los registros de los jugadores colombianos

Los operadores que pueden ser usados con la cláusula WHERE son:

OPERADOR	DESCRIPCION
=	Igual
<>	No igual // Diferente
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
BETWEEN	Entre un rango inclusivo
LIKE	Buscar un patrón
IN	Especificar múltiples posibles valores para una columna

5. AND, OR, NOT

SELECT columna1,columna2,…
FROM tabla;
WHERE condicion1 AND condicion2 AND condicion3 ….;

SELECT *full_name,club,age*
FROM *fifa1*
WHERE *nationality='Colombia'* AND *age<'23'*;

Se debe tener en cuenta que si el tipo de dato no está definido como entero y es un número, este debe de llevar comillas.

Con la consulta de ejemplo se obtienen los registros de los jugadores colombianos con menos de 23 años

SELECT column1,column2,…
FROM tabla;
WHERE condicion1 OR condicion2 OR condicion3 ….;

SELECT *full_name,club,age*
FROM *fifa1*
WHERE *club='Atletico Nacional Medellin'* OR *club='Juventus'*;

Con la consulta de ejemplo se obtienen los registros de los jugadores que pertenecen al Atlético Nacional o a la Juventus; es decir, las nomina completa de ambos equipos

SELECT column1,column2,…
FROM tabla;

WHERE NOT condicion;

SELECT *full_name,club,age*
FROM *fifa1*
WHERE NOT *age<'30'*;

Con la clausula NOT sese retorna lo contrario de la condición. En este caso la consulta retorna los registros de jugadores con edad mayor a 30

6. ORDER BY DESC, ASC

SELECT columna1,columna2,…
FROM tabla;
ORDER BY columna1,columna2,…;

SELECT *full_name,club,age,overall*
FROM *fifa1*
ORDER BY *age,overall, full_name*;

Con order by se toma condiciones para ordenar Ascendente o Descendentemente el resultado. Por defecto y si no se especifica nada el sistema toma un ordenamiento de menor a mayor inicialmente (Ascendentemente), ya sea de forma numérica o alfabética

En la consulta de ejemplo se toman 3 filtros de ordenamiento. Primero se ordena con base en la edad, luego al overall y por último el nombre.

SELECT *full_name,club,age,overall*
FROM *fifa1*
ORDER BY *age* DESC;

En esta consulta se especifica que el ordenamiento del resultado va a ser descendiente (de mayor a menor) basándose en la edad.

SELECT *full_name,club,age,overall*
FROM *fifa1*
ORDER BY *age* DESC, *overall* ASC;

En esta búsqueda se hace una combinación de orden Descendiente para la edad, y orden Ascendente para el overall. Es importante recordar la coma (,) después del primer filtro.

7. INSERT INTO

INSERT INTO nombre_tabla (columna1,column2a, columna3…)
VALUES (‘valor1’,’valor2’,’valor3’)

INSERT INTO *fifa1(full_name, club, age, league, overall,nationality)*
VALUES *('Tomas Maya','Atletico Nacional Medellin','21','Colombian Primera A','58',’Colombia’)*

Con la clausula Insert se insertan registros en la Base de datos.
En la sentencia de ejemplo se ingreso un jugador al Club Atlético Nacional

8. NULL Values

SELECT columna1 (,columna2,…)
FROM tabla;
WHERE columna IS NULL;

SELECT *full_name*
FROM *fifa1*
WHERE *club* IS NULL;

NULL corresponde a los campos de los registros que no tienen ningún valor asignado
En el ejemplo se seleccionan los jugadores que no pertenecen a ningún club

SELECT columna1 (,columna2,…)
FROM tabla;
WHERE columna IS NOT NULL;

SELECT *full_name*
FROM *fifa1*
WHERE *gk* IS NOT NULL;

Como la columna gk corresponde a Goalkeeper, esta busqueda retorna unicamente el nombre de todos los porteros

9. UPDATE

UPDATE nombre_tabla
SET columna1= 'valor', columna2= 'valor2'
WHERE condicion;

UPDATE *fifa1*
SET *club='Asociacion Deportivo Cali',overall=57*
WHERE *full_name='Ezequiel Palomeque'*;

Con update se hacen actualizaciones sobre los atributos de registros ya existentes.

Para el ejemplo se realizo una actualizacion para un jugador que cambio de club y que por su falta de continuidad disinuyo su rendimiento general.

Se debe tener cuidado con condiciones, ya que por ejemplo nationality=’Colombia’ puede pertenecer a varios registros y por ende se realizan múltiples cambios en todos los registros que cumplan con la condición. Y si no se coloca condición, el cambio se puede realizar para todos los registros

Los siguientes 2 ejemplos no se aplizaron en la base de datos pero pueden ser útiles para el aprendizaje:

*UPDATE fifa18
SET club='Asociacion Deportivo Cali',overall=57
WHERE nationality=’Colombia’; *
-Todos los registros con nationality=’Colombia’ quedarian con club='Asociacion Deportivo Cali' y overall=57

*UPDATE fifa18
SET club='Asociacion Deportivo Cali',overall=57*
-Todos los registros quedarian con club='Asociacion Deportivo Cali' y overall=57

10. DELETE

DELETE FROM nombre_tabla
WHERE condicion;

DELETE FROM *fifa1*
WHERE *full_name='Mauricio Molina';*

La sentencia DELETE sirve para eliminar registros.
En el ejemplo dado el jugador se ha retirado de su carrera futbolística y por ende no participara mas.

La siguiente declaración no se realizo pero es importante tenerla en cuenta

DELETE FROM nombre_tabla;
DELETE * FROM nombre_tabla;

La sentencia tambien sirve para eliminar toda la tabla. Se debe ser muy cauteloso con esta sentencia porque podría eliminar todos los registros sin posibilidad de recuperarlos

11. SELECT TOP-LIMIT

SELECT * FROM nombre_tabla LIMIT numero;

```
SELECT * FROM fifa1 LIMIT 5;

SELECT * FROM fifa1
WHERE nationality ='Colombia'
ORDER BY overall DESC
LIMIT 11;
```

Con la sentencia LIMIT se realiza una selección de los primeros registro de la búsqueda. En el primer ejemplo se seleccionan los 5 primeros registros de la tabla y en el segundo ejemplo se seleccionan los 11 jugadores de nacionalidad colombiana con mayor rendimiento general

12. MIN() & MAX()

```
SELECT MIN(nombre_columna)
FROM nombre_tabla
WHERE condicion ;

SELECT MAX(nombre_columna)
FROM nombre_tabla
WHERE condicion;
```

Con MIN y MAX hago la seleccion de unicamente el registro con el Minimo o Maximo valor de dicho atributo

```
SELECT MIN(overall), full_name
FROM fifa1
WHERE club ='Juventus' ;
```

Se selecciona el jugador con el menor overall de la Juventus

```
SELECT MAX(overall), full_name
FROM fifa1
WHERE nationality ='Colombia' ;
```

Se selecciona el jugador colombiano con el mayor overall

13. COUNT(), AVG() & SUM()

```
SELECT COUNT(nombre_columna)
FROM nombre_tabla
WHERE condicion ;

SELECT AVG(nombre_columna)
FROM nombre_tabla
WHERE condicion ;

SELECT SUM(nombre_columna)
FROM nombre_tabla
WHERE condicion ;
```

Realiza operaciones de conteo, promedio y suma en los campos de los registros; No necesariamente tienen que tener condición WHERE

```
SELECT COUNT(full_name),club
FROM fifa1
WHERE club='Juventus' ;
```

Cuenta el número de jugadores de la Juventus en el FIFA18

```
SELECT AVG(overall),club
FROM fifa1
WHERE club='Juventus' ;
```

Retorna el promedio del rendimiento general de los jugadores de la Juventus

```
SELECT SUM(age), club
FROM fifa1
WHERE club='Juventus' ;
```

Suma todas las edades de los jugadores de la Juventus.

14. LIKE

```
SELECT columna1, columna2, …
FROM nombre_tabla
WHERE columna  LIKE patron;
```

LIKE se utiliza para encontrar patrones específicos en una columna.
Hay dos símbolos comodines usados en conjunto con el operador LIKE: '%', '_'

```
SELECT full_name,  nationality
FROM fifa1
WHERE club  LIKE '%Madrid%';
```

Retorna el nombre y el quipo de los jugadores que tienen la palabra ‘Madrid’ en el nombre del club (Atletico de Marid y Real Madrid)

```
SELECT full_name,  nationality
FROM fifa1
WHERE full_name  LIKE '%guez';
```

Retorna el nombre y la nacionalidad de todos los jugadores que su apellido finaliza en ‘guez’; es conveniente ya que hay apellidos como Rodriguez, que están escritos con tilde y sin tilde en la i; también arroja otros jugadores con apellidos como Domínguez, Mínguez, Miguez, Iñiguez.

```
SELECT full_name, nationality,age
FROM fifa1
WHERE full_name  LIKE 'Diego%';
```

Retorna todos los jugadores con nombre ‘Diego’ ; es muy probable por el gran numero de fanáticos Maradona

```
SELECT full_name,  nationality,age
FROM fifa1
WHERE full_name  LIKE '_do%';
```

Selecciona los jugadores que en su nombre (NO apellido) las segunda y tercera letra son do = Adolfo, Edoardo, Adolphe , Ado, Edon, Adonis

```
SELECT full_name,  nationality,age
FROM fifa1
WHERE full_name  LIKE 'Crist_%';
```

Selecciona los jugadores que sus nombres empiezan con ‘Crist’ y que al menos tienen 6 letras = Cristiano,Cristian, Cristhian, Cristofer

```
SELECT full_name,  nationality,age
FROM fifa1
WHERE full_name  LIKE 'L%l';
```

Selecciona los jugadores que sus nombres empiezan con ‘L’ y que apellido en ‘l’ = Luis Muriel , Lars Stindl, Ludovix Baal.

COMODINES

Operador LIKE	Descripción
‘ a%’	Encuentra los valores que empiezan con ‘a’
‘ %a’	Encuentra los valores que terminan con ‘a’
‘ %or%’	Encuentra los valores que tienen ‘or’ en cualquier posición
‘ _r’	Encuentra los valores que tienen ‘r’ en la segunda posición
‘ a_%_%’	Encuentra los valores que empiezan con ‘a’ y tienen al menos 3 caracteres de longitud
‘ a%o’	Encuentra los valores que empiezan con ‘a’ y terminan con ‘o’

SELECT *full_name, nationality, club*
FROM *fifa1*
WHERE *club* LIKE '*Atl*%';
Retrona los jugadores con nombre y nacionalidad de quipos que comienzan con Atl (Atlético, Atletico)

SELECT *full_name, nationality, club*
FROM *fifa1*
WHERE *club* LIKE '%*City*%';
Retorna los jugadores con nombre y nacionalidad de quipos que tienen la palabra City en el nombre (Manchester city, Hull city, Stoke City, Swansea City, Leicester City)

SELECT *full_name, nationality, club*
FROM *fifa1*
WHERE *nationality* LIKE '___*land*';
Retorna los jugadores con nombre y nacionalidad de Países de 7 letras que terminan en ‘land’ (England, Iceland)

SELECT *full_name, nationality, club*
FROM *fifa1*
WHERE *nationality* LIKE '_*a_a_a*';
Retorna los jugadores con nombre y nacionalidad de Paises de 6 letras que tienen 3 ‘a’ intercaladas entre consonantes '_a_a_a' (Canada, Panama)

15. IN

```
SELECT nombre_columna(s)
FROM tabla
WHERE nombre_columna IN (valor1,valor2,…);
```

Este operador permite especificar múltiples valores para la cláusula WHERE.
Este operador es un acortador para múltiples condiciones OR

```
SELECT full_name,  nationality, club
FROM fifa1
WHERE nationality IN ('Colombia', 'Brazil');
```
Retorna los jugadores con nombre y nacionalidad de Colombia O Brazil ; en este caso la cláusula IN funciona casi como un '&'.

```
SELECT full_name,  nationality, club, age
FROM fifa1
WHERE age NOT IN ('20', '21','22', '23','24', '25','26', '27','28', '29');
```
Retorna los jugadores con edad menores de 20 y mayores de 29

```
SELECT full_name,  nationality, club, age, overall
FROM fifa1
WHERE overall
IN ( SELECT overall FROM fifa1 WHERE nationality='Colombia'  AND age='25' AND
club='FC Bayern Munich');
```

En este caso se presenta una búsqueda anidada. Primero se hace la búsqueda interna en la que se selecciona el overall del jugador colombiano de 25 años que juega en el FC Bayer munich (James Rodriguez), y luego selecciona los jugadores con nombre, nacionalidad, club y edad que tienen el mismo overall al que seleccióno en la primera búsqueda.

La consulta interna (el segundo FROM) se puede utilizar para buscar en otra tabla diferente.

```
SELECT * FROM tabla
WHERE columnaX IN (SELECT columnaX FROM  tabla2)
```

16. BETWEEN

```
SELECT nombre_columna(s)
FROM tabla
WHERE nombre_columna BETWEEN valor1 AND valor2;
```

```
SELECT full_name,  nationality, club, age, overall
```

```
FROM fifa1
WHERE age BETWEEN '16' AND '19';
```

Retorna los jugadores con edad entre 16 y 19 años (incluye los de 16 y 19)

```
SELECT full_name,  nationality, club, age, overall
FROM fifa1
WHERE age NOT BETWEEN '22' AND '32';
```

Selecciona los jugadores con edad menores de 21 y mayores de 33 años (elimina los jugadores con edad entre el rango de 22 y 32 años)

```
SELECT full_name,  nationality, club, age, overall
FROM fifa1
WHERE (age BETWEEN '16' AND '19')
AND NOT nationality IN ('Chile','Colombia','Argentina','Brazil')
```

Retorna los jugadores entre 16 y 19 años que su nacionalidad NO es Chile, Colombia, Brazil, Argentina

```
SELECT full_name,  nationality, club, age, overall
FROM fifa1
WHERE (age BETWEEN '16' AND '19')
AND  nationality IN ('Chile','Colombia','Argentina','Brazil')
AND overall>70
```

Retorna los jugadores entre 16 y 19 años que su nacionalidad es Chile, Colombia, Brazil, Argentina (No hay jugadores chilenos que cumplan con la condicion)

Cuando el BETWEEN se utiliza con string-cadena de caracteres entonces los ordena alfabéticamente

```
SELECT full_name,  nationality, club, age, overall
FROM fifa1
WHERE nationality BETWEEN 'Brazil' AND 'Colombia'
ORDER BY nationality;
```

El resultado arroja jugadores de los paises : Brazil, Burkina Faso, Cameroon, Canada, Cape Verde, Central Africa Rep., Çhile y Colombia

```
SELECT full_name,  nationality, club, age, overall
FROM fifa1
WHERE nationality NOT BETWEEN 'Brazil' AND 'Ukraine'
ORDER BY nationality;
```

Por el limite inferior retorna paises como Albania , Algeria, Argentina.. y por el limite superior retorna paises como Unites States, Uruguay y Wales
Forma alterna del BETWEEN:

```
SELECT DISTINCT full_name,club,age,league,overall
FROM fifa1
WHERE nationality='Colombia' AND age >'18' AND age<'20';
```

17. ALIASES AS

Son Usados para dar a una tabla , o una columna de una tabla , un nombre temporal

Alias columna

```
SELECT nombre_columna AS alias
FROM tabla;
```

```
SELECT nationality AS na
FROM fifa1;
```

El resultado retorna todas las nacionalidades sin repetir

Alias table

```
SELECT nombre_columna(s)
FROM tabla AS alias;
```

```
SELECT full_name, age,overall
FROM fifa1 AS ff;
```

Para multiples Alias

```
SELECT nombre_columna(s)
FROM tabla1 AS alias1, tabla2 AS alias2
WHERE alias1.nombrecolumna = 'XYZ' AND  alias1.columnaNUEVA =
alias2.nombrecolumna;
```

Se puede usar para columnas de mismo nombre en diferentes tablas

18. JOIN

```
SELECT tabla1.columnaX, tabla2.columnaY
FROM tabla1
INNER JOIN tabla2 ON
tabla1.columnaZ = tabla2.columnaZ;
```

Esta cláusula es usada para combinar resultados de campos o filas de dos o más tablas, basado en una relación entre las columnas

```
SELECT fifa1.full_name, fifa2.age
FROM fifa1
INNER JOIN fifa2 ON
fifa1.ID = edad_act.ID;
```

En la tabla número uno se presenta un valor de la edad desactualizada; y en la tabla número dos se presentan la edad actualizada junto a otros atributos de los jugadores. Ambas tablas comparten algunos atributos iguales, y se tomo como referencia la columna ID que es la que tiene un valor único para cada jugador.
Esta consulta arroja el nombre (obtenido de la tabla1) y la edad actualizada de la tabla2

EXISTEN DIFERENTES TIPOS DE SQL JOINs

(INNER) JOIN: Retorna registros que tienen valores 'matching' (atributos iguales, que coinciden) , en ambas tablas
LEFT (OUTER) JOIN: Retorna todos los registros de la tabla izquierda , y los registro emparejados (coincidentes) de la tabla derecha

INNER JOIN

```
SELECT nombre_columna(s)
FROM tabla1
INNER JOIN tabla2 ON tabla1.nombre_columna = tabla2.nombre_columna;
```

```
SELECT fifa1.full_name, edad_act.age
FROM fifa1
INNER JOIN edad_act ON fifa1.ID = fifa2.ID;
```

Toma el nombre de la primera tabla, busca el emparejamiento de la columna ID y selecciona la edad (columna age) de la segunda tabla

Igualmente se puede unir a una Tercera tabla

```
SELECT nombre_columna(s)
FROM (( tabla1
INNER JOIN tabla2 ON tabla1.nombre_columnax = tabla2.nombre_columnax)
INNER JOIN tabla3 ON tabla1.nombre_columnay = tabla3.nombre_columnay )) ;*
```

LEFT JOIN

```
SELECT nombre_columna(s)
FROM tabla1
LEFT JOIN tabla2 ON tabla1.nombre_columna = tabla2.nombre_columna;
```

```
SELECT fifa1.full_name, fifa2.age
FROM fifa1
LEFT JOIN fifa2 ON
fifa1.ID = fifa2.ID
ORDER BY fifa2.age;
```

Retorna el mismo Resultado del INNER JOIN + el jugador añadido que no tiene nombre en la segunda tabla. Toma todos los datos de la izquierda (full_name) y los une con su correpondiente (en caso que tenga) en la tabla de la derecha.

19. UNION

Se usa para combinar el conjunto-resultado de dos o más declaraciones SELECT
Cada declaración SELECT dentro de UNION debe tener el mismo número de columnas
Las columnas deben tambien tipos de datos similares
Las columnas en cada declaración SELECT debe tener también el mismo orden.

```
SELECT nombre_columna(s) FROM tabla1
UNION
SELECT nombre_columna(s) FROM tabla2;
```

```
SELECT nombre_columna(s) FROM tabla1
UNION ALL
SELECT nombre_columna(s) FROM tabla2;
```

```
SELECT nombre_columna(s) FROM tabla1
WHERE condicion
UNION ALL
SELECT nombre_columna(s) FROM tabla2
WHERE condicion
ORDER BY;
```

```
SELECT age  FROM fifa1
UNION
SELECT age FROM fifa2
ORDER BY age;
```

Une todos los valores de ambas columnas de tablas diferentes; elimina los valores iguales y los ordena.

20. GROUP BY

```
SELECT nombre_columna(s)
FROM tabla
WHERE condición
GROUP BY nombre_columna(s)
ORDER BY nombre_columna(s);
```

SELECT COUNT(*full_name*), *nationality*
FROM *fifa1*
GROUP BY *nationality*;
Agrupa y cuenta todos los jugadores según su nacionalidad

SELECT COUNT(*full_name*), *nationality*
FROM *fifa1*
GROUP BY *nationality*
ORDER BY COUNT(*full_name*) DESC;
Agrupa y cuenta todos los jugadores según su nacionalidad y los ordena de mayor a menor según la cantidad por nacionalidad

SELECT *fifa1.league*, COUNT(*fifa1.full_name*) AS *num_jugs* , AVG(*fifa2.age*) AS *prom_edad*
FROM fifa1
LEFT JOIN f*ifa2* ON *fifa1.ID* = *fifa2.ID*
GROUP BY *fifa1.league*;
Retorna el número total de jugadores en cada liga y el promedio de edad

21. HAVING

La cláusula HAVING fue añadida a SQL porque la palabra clave WHERE no podía ser usada con funciones agregadas

SELECT nombre_columna(s)
FROM tabla
WHERE condición
GROUP BY nombre_columna(s)
HAVING condition
ORDER BY nombre_columna(s);

SELECT COUNT(*full_name*), *league*
FROM *fifa1*
GROUP BY *league*
HAVING COUNT(*full_name*) >500
ORDER BY COUNT(*full_name*);

Retorna las ligas con mas de 500 jugadores. El ORDER BY se puede omitir.
SELECT *fifa1.league*, COUNT(*fifa2.age*)
FROM f*ifa1*
INNER JOIN *fifa2* ON *fifa1.ID* = *fifa2.ID*
WHERE *fifa2.age* <'*23*'
GROUP BY *fifa1.league*
HAVING COUNT(*fifa2.age*)>*200*;

Esta consulta retorna las ligas con más de 200 jugadores que tienen una edad menor a 23 años (no incluye la edad de 23). Notar que COUNT(fifa2.age) es en sí COUNT(fifa2.age < '23'), solo tiene en cuenta los jugadores menores de 23 años

22. CREATE DB

```
CREATE DATABASE nombre_db;
```

//Para sqlite manager se debet crear desde la barra de menu, Base de Datos, nueva Base de Datos
// Cuando se crea una base de Datos en SQLite Manager se cierra la otra, solo puede trabajar con una a la vez (y con temporales)

23. DROP DB

```
DROP DATABASE nombre_db;
```

// Tecnicamente en SQLite Manager NO existe CREATE DATABASE y tampoco DROP DATABASE

24. DETACH DB

DETACH DATABASE Colombia;
// Se usa para desconectar la base de datos, pero no esta disponible en SQLite Manager

25. CREATE TABLE

```
CREATE TABLE nombre_table(
    Columna1 datatype,
    Columna2 datatype,
        . . . . .
    );
```

Crear tablas nuevas

```
CREATE TABLE Jugadores_Nacionales (
nombre VARCHAR,
edad INTEGER
);
```

No puede haber tablas con nombres iguales

```
CREATE TABLE Colombianos AS
SELECT full_name, league
FROM fifa1
WHERE nationality= 'Colombia'
ORDER BY league;
```

Se crea una nueva tabla, con el nombre y la liga, de jugadores colombianos solamente.

26. DROP TABLE

Se utiliza para eliminar tablas; hay que ser cuidadosos porque la información no se puede recuperar

```
DROP TABLE nombre_tabla;
```

DROP TABLE Jugadores_Nacionales;

27. ALTER TABLE

Es usado pasa agregar, eliminar, o modificar columnas en una tabla existente
También es usado para agregar o eliminar restricciones en una tabla existente

```
ALTER TABLE nombre_tabla
ADD nombre_columna tipo_dato;
```

```
ALTER TABLE Colombianos
ADD Potencial INTEGER;
```

Agrega una columna llamada Potencial de tipo de dato entero

```
ALTER TABLE Colombianos
RENAME TO Jugadores_COL;
```

Renombrar una tabla

28. CONSTRAINTS (RESTRICCIONES)

Son usadas para especificar las reglas para los datos en las tablas
Se utilizan para limitar el tipo de dato que se puede agregar en la tabla; esto asegura la precisión y confianza en los datos de la tabla. Si hay alguna violación entre las restricciones y los datos en acción, la acción es abortada.

RESTRICCION	DESCRIPCION
NOT NULL	Asegura que una columna no pueda tener un valor NULL
UNIQUE	Asegura que todos los valores en una columna son diferentes
PRIMARY KEY	Una combinación de un NOT NULL and UNIQUE. Identifica de manera particular/singular cada fila en una tabla.
FOREIGN KEY	Identificador singular/único de una fila/registro en otra tabla
CHECK	Asegura que todos los valores en una columna satisfacen una condición especifica
DEFAULT	Establece un valor determinado para una columna cuando no se especifica ningún valor.
INDEX	Se usa para crear y recuperar datos de la base de datos muy rapidamente

```
CREATE TABLE nombre_table(
 Columna1 datatype RESTRICCION ,
 Columna2 datatype RESTRICCION,
  . . . . .
   );
```

```
CREATE TABLE Jugadores_Nacional (
  nombre VARCHAR PRIMARY KEY,
  edad INTEGER NOT NULL
);
```

NOT NULL

NOT NULL impone/fuerza a que una columna no acepte valores NULL
Esto impone a un campo a siempre contener un valor, lo que significa que no se puede insertar un nuevo registro, o actualizar un registro sin agregar un valor a dicho campo.

```
CREATE TABLE Jugadores_Nuevos (
nombre VARCHAR NOT NULL,
edad INTEGER NOT NULL,
potencial INTEGER NOT NULL
);
```

UNIQUE

Asegura que todos los valores en una columna son diferentes
Tanto el UNIQUE y la PRIMARY KEY son restricciones que proveen garantía para unicidad en una columna o un conjunto de columnas
La restricción PRIMARY KEY automáticamente tiene una restricción UNIQUE
Sin embargo se pueden tener varias restricciones UNIQUE en una tabla, pero solo una restricción PRIMARY KEY por tabla.

```
CREATE TABLE club_nuevos (
ID_Jugadores INTERGER NOT NULL UNIQUE,
club VARCHAR NOT NULL,
edad INTEGER NOT NULL,
potencial INTEGER NOT NULL
);
```

PRIMARY KEY

Esta restricción para tener un identificador único para cada registro en la tabla de la base de datos
Las Primary keys deben contener valores UNIQUE, y no pueden contener valores NULL
Una tabla solo puede tener una PRIMARY KEY, la cual puede consistir en una columna o varias

```
CREATE TABLE club_nuevos2 (
ID_Jugadores INTERGER NOT NULL UNIQUE,
club VARCHAR NOT NULL,
edad INTEGER NOT NULL,
potencial INTEGER NOT NULL,
PRIMARY KEY (ID_Jugadores)
);
```

FOREIGN KEY

Una FOREIGN KEY es una llave usada para enlazar dos tablas juntas.
Una FOREIGN KEY es una columna (o colección de columnas) en una tabla que se refieren/remiten a la PRIMARY KEY en otra tabla.
La tabla que contiene la clave externa se denomina tabla secundaria y la tabla que contiene la clave candidata se denomina principal o referenciada.

```
CREATE TABLE club_nuevos3 (
ID_Jugadores INTERGER  NOT NULL UNIQUE,
club VARCHAR NOT NULL,
edad INTEGER NOT NULL,
potencial INTEGER NOT NULL,
ID,
PRIMARY KEY (ID_Jugadores),
FOREIGN KEY (ID) REFERENCES fifa1(ID)
);
```

CHECK

Esta restricción es usada para limitar el rango de valores que pueden ser colocados en una columna
Si se define la restricción CHECK en una columna particular esto permite solamente ciertos valores para esa columna
Si se defina la restricción CHECK en una tabla esto puede limitar los valores en ciertas columnas basado en valores en otras columnas en una fila

```
CREATE TABLE club_nuevos4 (
ID_Jugadores INTERGER  NOT NULL UNIQUE,
club VARCHAR NOT NULL,
edad INTEGER NOT NULL,
potencial INTEGER NOT NULL,
CHECK (edad>=16)
);
```

Las Restricciones se pueden alterar con ALTER COLUMN : añadir (ADD), eliminar (DROP) y modificar (MODIFY) dependiendo del sistema de gestión de bases de datos que se esté utilizando.

DEFAULT

Esta restricción es usada para proveer un valor por defecto a alguna columna

```
CREATE TABLE Jugadores_nuevos5 (
ID_Jugadores INTERGER  NOT NULL UNIQUE,
club VARCHAR NOT NULL,
edad INTEGER NOT NULL,
nacionalidad DEFAULT 'Colombia'
);
```

INDEX

Esta declaración es usada para crear índices en las tablas.
Los índices son usados para recuperar, retornar datos de una base de datos de manera muy rápida. Los usuarios no pueden ver los índices, solo son usados para acelerar las búsquedas/consultas.

NOTA: actualizar una tabla con índices toma más tiempo que actualizar una tabla sin índice (porque los índices también necesitan una actualización). Por lo tanto solo es adecuado crear índices en columnas que se van a buscar con frecuencia.

```
CREATE INDEX nombre_indice
ON table (nombre columna(s));
```

```
CREATE INDEX id_name
ON fifa2(ID);
```

```
CREATE UNIQUE INDEX nombre_indice
    ON table (nombre columna(s));
```

AUTO INCREMENT

El autoincremento permite un único número ser generado automáticamente cuando un nuevo registro es insertado dentro de una tabla.
A menudo este es un campo de llave primaria que nos gustaría fuera creado automáticamente cada vez que un nuevo registro es insertado

```
CREATE TABLE club_nuevos6 (
ID INTERGER  PRIMARY KEY ,
nombre VARCHAR
);
```

INTERGER & PRIMARY KEY : al tener estas dos condiciones para una sola columna se genera una condición llamada ROWID, que es implícitamente autoincremental

29. DATES

Cada Sistema de Gestion de Bases de Datos (SGDB) soporta tipos de datos diferente, con sus respectivas funciones asociadas

SQLite soporta 5 tipos

FUNCION	FORMATO RETORNADO
Date	YYYY-MM-DD
Time	HH:MM:SS
Datetime	YYYY-MM-DD HH:MM:SS
Julianday	Basado en meridiano de Greenwich
strftime	Formato string

SELECT * FROM *fifa1* WHERE *birth_date ='05/02/1986'*;
Se seleccionan los jugadores que nacieron el *'05/02/1986'*
Se debe ser detallista, con el orden adecuado de día, mes y año

30. VIEWS

En SQL una VIEW (vista) es una tabla virtual basada en el resultado o conjunto de resultados de una declaración/búsqueda SQL
Son guardadas en la base de datos para ser consultadas posteriormente

CREATE VIEW *futbol_veterano* AS
SELECT *club,full_name*
FROM *fifa1*
WHERE *age* >'*33*';

Luego, para su posterior visualización se puede utilizar:

SELECT * FROM [*futbol_veterano*];

A la posterior visualización de la búsqueda tambien se le puede añadir WHERE. Las VIEWS se pueden REPLACE y DROP.

31. INJECTION

SQL injection es una técnica de inyección de código que podría destruir una base de datos. Es una de las técnicas de hackeo web más comunes. Consiste en la colocación de código malicioso en las declaraciones de SQL, a través de los INPUT de la página web. En lugar de rellenar adecuadamente campos como usuario y contraseña, se colocan sentencias SQL que alteran la base de datos.

32. HOSTING DB

Los servidores web deben tener acceso a un sistema de gestión de base de datos que use SQL; así se puede almacenar y recuperar datos de la bases de datos de un sitio web.

33. TRIGGER

Desencadenador/Disparador
Es una porción de código que se va a disparar cuando se haga una acción en una tabla (INSERT,UPDATE,DELETE)
FOR ,AFTER, INSTEAD OF → indican en que momento se quiere activar el Trigger y después colocamos la acción (INSERT,UPDATE,DELETE)

Son porciones de código que se activan luego de realizar cierta acción en una tabla.

CREATE TRIGGER nombre_trigger AFTER/BEFORE
INSERT/UPDATE/DELETE ON tabla
BEGIN
“fragment de Codigo”
END;

```
CREATE TRIGGER triger_1 AFTER INSERT ON fifa2
BEGIN
  SELECT * FROM fifa2;
END;
```

Referencias

https://www.w3resource.com/sqlite/
https://www.w3schools.com/sql/
https://sqlite.org