CODERHOUSE

Proyecto Final – Kings League

Segunda entrega
Ivan Felipe Chavez Cortes
Tutor: Leonel Lo Presti
Curso de SQL

1. Vistas

Vista 1: Estadísticas Jugadores

```sql
CREATE OR REPLACE VIEW estadisticas_jugadores AS
    (SELECT
        CONCAT(j.nombre, ' ', j.apellido) Nombre,
        IFNULL(g.cantidad_goles, 0) Cantidad_Goles,
        IFNULL(a.cantidad_asistencias, 0) Cantidad_Asistencias,
        e.nombre AS Equipo
    FROM
        jugador j
            LEFT JOIN
        (SELECT
            id_jugador, COUNT(*) cantidad_goles
        FROM
            goles
        GROUP BY id_jugador) g ON j.id_jugador = g.id_jugador
            LEFT JOIN
        (SELECT
            id_jugador, COUNT(*) cantidad_asistencias
        FROM
            asistencias
        GROUP BY id_jugador) a ON j.id_jugador = a.id_jugador
            JOIN
        equipo e ON e.id_equipo = j.id_equipo
    WHERE
        g.cantidad_goles > 0
            OR a.cantidad_asistencias > 0
    ORDER BY g.cantidad_goles DESC);
```

Descripción: En esta vista pueden ser consultados los goles y asistencias de cada jugador, se encuentra ordenada por la cantidad de goles.

Objetivo: El objetivo de esta vista es poder consultar de forma sencilla y rápida las estadísticas de cada uno de los jugadores en lo que a goles y asistencias hace referencia, adicionalmente se excluyen a los jugadores que no hayan marcado goles o asistencias con el fin de filtrar la información.

Tablas/Datos: Jugador, Goles, Asistencias y Equipo

Vista 2: Jugadores Franquicia

```
CREATE OR REPLACE VIEW jugadores_franquicia AS
    (SELECT
        CONCAT(j.nombre, ' ', apellido) Nombre_Jugador_Franquicia,
        e.nombre Equipo
    FROM
        jugador j
            JOIN
        equipo e ON e.id_equipo = j.id_equipo
    WHERE
        franquicia = 1);
```

Descripción: En esta vista se pueden conocer los nombres y apellidos de todos los jugadores franquicia y el equipo al que pertenecen.

Objetivo: El objetivo de esta vista es permitir de forma rápida y sencilla conocer los jugadores franquicia que participan en la liga, adicionalmente conocer a qué equipo pertenecen sin necesidad de realizar consultas u operaciones adicionales.

Tablas/Datos: Equipo y Jugador

Vista 3: Detalle Partidos

```
CREATE OR REPLACE VIEW detalle_partidos AS
    (SELECT
        CONCAT(l.nombre, '-', v.nombre) Equipos,
        CONCAT(p.goles_local, '-', goles_visitante) Goles_Partido,
        fecha Fecha
    FROM
        partido p
            JOIN
        disputa d ON p.id_partido = d.id_partido
            JOIN
        equipo l ON d.id_local = l.id_equipo
            JOIN
        equipo v ON d.id_visitante = v.id_equipo
    ORDER BY fecha DESC);
```

Descripción: En esta vista pueden ser consultados el resultado de los partidos, los equipos enfrentados y adicionalmente son ordenados por fecha.

Objetivo: El objetivo de esta vista es poder conocer el resultado final de cada uno de los partidos disputados en el transcurso de la liga, el nombre de los equipos enfrentados y la fecha en la que se disputó el partido, todo con el fin de consultar esta información de forma rápida, breve y resumida. Esta vista puede ser útil al momento de ejecutar resúmenes de las jornadas o highlights.

Tablas/Datos: Partido, Disputa y Equipo

Vista 4: Detalle Patrocinios

```
CREATE OR REPLACE VIEW detalle_patrocinios AS
    (SELECT
        e.nombre Equipo,
        p.nombre Patrocinador,
        CONCAT('€ ', p.dinero_invertido) Dinero_Invertido
    FROM
        equipo e
            JOIN
        patrocinador p ON e.id_equipo = p.id_equipo
    ORDER BY p.dinero_invertido DESC);
```

Descripción: En esta vista se pueden consultar los detalles de los patrocinadores de cada uno de los equipos incluyendo: nombre, presupuesto y equipo patrocinado. Adicionalmente, la información es ordenada de acuerdo al presupuesto invertido por cada uno.

Objetivo: El objetivo de esta vista es poder visualizar de forma breve y resumida la información de cada uno de los patrocinadores junto con el equipo patrocinado y el presupuesto invertido, esta información puede ser consultada por equipos de ética y competencia para garantizar que los patrocinadores cumplan con los rangos presupuestales adicionalmente esta información puede ser utilizada por el equipo contable o de impuestos.

Tablas/Datos: Equipo y Patrocinador

Vista 5: Cantidad Partidos Arbitrados

```
CREATE OR REPLACE VIEW cantidad_partidos_arbitros AS
    (SELECT
        CONCAT(nombre, ' ', apellido) Nombre_Arbitro,
        COUNT(p.id_arbitro) Cantidad_Partidos
    FROM
        arbitro a
            JOIN
        partido p ON a.id_arbitro = p.id_arbitro
    GROUP BY a.id_arbitro
    ORDER BY Cantidad_Partidos DESC);
```

Descripción: En esta tabla es posible consultar la cantidad de partidos arbitrados por cada uno de los integrantes del cuerpo arbitral.

Objetivo: El objetivo de esta tabla es llevar un conteo y control de los partidos arbitrados por cada integrante del cuerpo arbitral a lo largo de la temporada.

Tablas/Datos: Arbitro y Partido

2. Funciones

Función 1: Mayor Goleador Asistidor

```
DROP FUNCTION IF EXISTS mayor_goleador_asistidor;
DELIMITER //
CREATE FUNCTION mayor_goleador_asistidor(opcion VARCHAR(20)) RETURNS VARCHAR(100)
DETERMINISTIC
BEGIN
    DECLARE resultado VARCHAR(100);
    CASE
        WHEN opcion LIKE 'goleador' THEN
            SELECT GROUP_CONCAT(nombre SEPARATOR ', ')
            INTO resultado
            FROM estadisticas_jugadores
            WHERE cantidad_goles = (SELECT MAX(cantidad_goles) FROM estadisticas_jugadores);
        WHEN opcion LIKE 'asistidor' THEN
            SELECT GROUP_CONCAT(nombre SEPARATOR ', ')
            INTO resultado
            FROM estadisticas_jugadores
            WHERE cantidad_asistencias = (SELECT MAX(cantidad_asistencias) FROM estadisticas_jugadores);
        ELSE
            SET resultado = 'Por favor ingresa un valor válido, las opciones son: Goleador - Asistidor. Gracias!';
    END CASE;
    RETURN resultado;
END //
DELIMITER ;
```

Descripción: Esta función permite ingresar opciones o parámetros (goleador y asistidor) y retorna el nombre del jugador que más goles o asistencias ha realizado hasta la fecha, de acuerdo con la opción seleccionada. Adicionalmente, tiene una gestión en caso de errores con el parámetro ingresado.

Objetivo: El objetivo de esta función es obtener los nombre(s) del máximo goleador(es), para que cualquier persona pueda tener acceso a la información de los jugadores más destacados y con mejor rendimiento de la liga.

Tablas/Datos: Vista estadisticas_jugadores.

Función 2: Buscador Id Jugador

```
DROP FUNCTION IF EXISTS buscador_id_jugador;
DELIMITER $$
CREATE FUNCTION buscador_id_jugador(opcion INT) RETURNS VARCHAR(100)
DETERMINISTIC
BEGIN
DECLARE resultado_equipo VARCHAR(30);
DECLARE nombre_jugador VARCHAR(100);

SELECT
    e.nombre
INTO resultado_equipo FROM
    jugador j
        JOIN
    equipo e ON j.id_equipo = e.id_equipo
WHERE
    j.id_jugador = opcion;
SELECT
    CONCAT(j.nombre, ' ', j.apellido)
INTO nombre_jugador FROM
    jugador j
        JOIN
    equipo e ON j.id_equipo = e.id_equipo
WHERE
    j.id_jugador = opcion;
    IF resultado_equipo IS NOT NULL THEN
    RETURN CONCAT('El jugador ',nombre_jugador,' pertenece a: ',resultado_equipo);
    ELSE
    RETURN CONCAT('No se encontró un jugador dentro de la Kings League con el ID ingresado: ', opcion);
    END IF;
END $$
DELIMITER ;
```

Descripción: Esta función permite conocer el nombre y el equipo al que pertenece un jugador con tan solo ingresar su ID único. Adicionalmente, tiene una gestión en caso de errores con el parámetro ingresado.

Objetivo: El objetivo de esta función es ser utilizada como un buscador únicamente ingresando el ID del jugador, permitiendo que cualquier fanático pueda conocer la información principal de cada uno de los jugadores, adicionalmente también podría ser utilizada por empleados dentro de la Kings League.

Tablas/Datos: Jugador

Función 3: Buscador Nombre Jugador

```
DROP FUNCTION IF EXISTS buscador_nombre_jugador;
DELIMITER $$
CREATE FUNCTION buscador_nombre_jugador(nombre varchar(50)) RETURNS VARCHAR(100)
DETERMINISTIC
BEGIN
DECLARE resultado_equipo VARCHAR(30);
DECLARE nombre_jugador VARCHAR(100);

SELECT
    e.nombre
INTO resultado_equipo FROM
    jugador j
        JOIN
    equipo e ON j.id_equipo = e.id_equipo
WHERE
    CONCAT(j.nombre, ' ', j.apellido) LIKE nombre;
SELECT
    CONCAT(j.nombre, ' ', j.apellido)
INTO nombre_jugador FROM
    jugador j
        JOIN
    equipo e ON j.id_equipo = e.id_equipo
WHERE
    CONCAT(j.nombre, ' ', j.apellido) LIKE nombre;
    IF resultado_equipo IS NOT NULL THEN
    RETURN CONCAT('El jugador ',nombre_jugador,' pertenece a: ',resultado_equipo);
    ELSE
    RETURN CONCAT('No se encontró un jugador dentro de la Kings League con el nombre ingresado: ', nombre);
    END IF;
END $$
DELIMITER ;
```

Descripción: Esta función permite conocer el nombre y el equipo al que pertenece un jugador con tan solo ingresar su nombre y apellido. Adicionalmente, tiene una gestión en caso de errores con el parámetro ingresado.

Objetivo: El objetivo de esta función es ser utilizada como un buscador únicamente ingresando el nombre y apellido del jugador, permitiendo que cualquier fanático pueda conocer la información principal de cada uno de los jugadores, adicionalmente también podría ser utilizada por empleados dentro de la Kings League.

Tablas/Datos: Jugador

3. Stored Procedures

Procedure 1: Orden Jugadores

```
DROP PROCEDURE IF EXISTS orden_jugadores;
DELIMITER //
CREATE PROCEDURE orden_jugadores(
    IN campo VARCHAR(20),
    IN orden VARCHAR(20)
)
BEGIN
    IF orden = 'ascendente' THEN
        SET orden = 'ASC';
    ELSEIF orden = 'descendente' THEN
        SET orden = 'DESC';
    ELSE
        SET orden = 'DESC';
    END IF;
    ##Aqui se aplica el uso de clausulas para efectar el SP
    SET @jugadores = CONCAT('SELECT * FROM jugador ORDER BY ', campo, ' ', orden);
    PREPARE stmt FROM @jugadores;
    EXECUTE stmt;
    DEALLOCATE PREPARE stmt;
END //
```

Descripción: Este Stored Procedure permite ordenar el listado de todos los jugadores de la liga de acuerdo al campo de ordenamiento y el tipo de ordenamiento ingresado.

Objetivo: Tiene como objetivo obtener de manera rápida y sencilla la lista ordenada de jugadores ascendente o descendente de acuerdo al campo requerido en cada caso de uso.

Tablas/Datos: Jugador

Procedure 2: Insert Patrocinador

```
DROP PROCEDURE IF EXISTS sp_insert_patrocinador;
DELIMITER $$
CREATE PROCEDURE sp_insert_patrocinador(IN nombre VARCHAR(50), IN dinero INT, IN equipo INT,OUT output VARCHAR(100))
BEGIN
    IF (nombre <> '') and (dinero and equipo != 0) THEN
        INSERT INTO PATROCINADOR (id_patrocinador,nombre,dinero_invertido,id_equipo) VALUES (null,UCASE(nombre),dinero,equipo);
        SET output = 'Inserción del patrocinador exitosa.';
    ELSE
        SET output = 'ERROR: no se pudo registrar al patrocinador del equipo.';
    END IF;
END
$$
```

Descripción: Este Stored Procedure facilita la inserción de registros de patrocinadores, adicionalmente genera respuestas de acuerdo al resultado de la inserción.

Objetivo: Tiene como objetivo facilitar la inserción de registros a la tabla Patrocinador, únicamente requiere la información y por defecto genera la sintaxis e ingresa los registros.

Tablas/Datos: Patrocinador

4. Archivos SQL

4.1 Script creación de objetos:

Link a archivo .sql: enlace.

4.2 Script inserción de datos:

Link a archivo .sql: enlace.