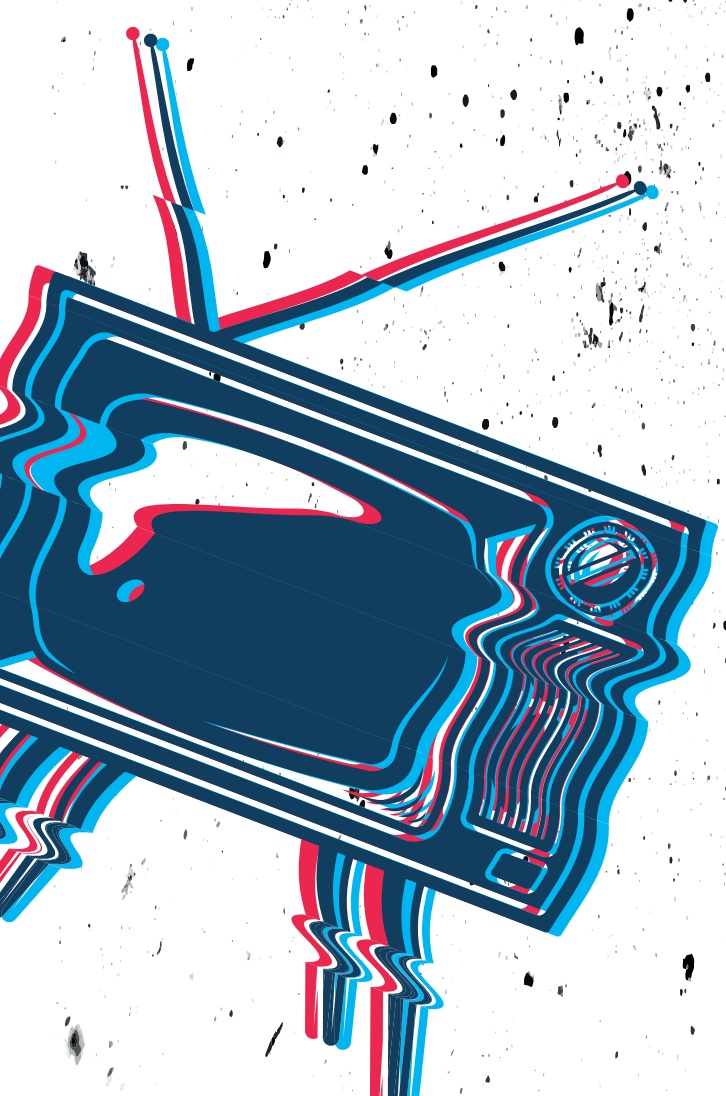
43405

CODERHOUSE

PRE-ENTREGA FINAL

RECHE JOSÉ



Proyecto final
segunda entrega

VISTAS

Primer vista: Partidos rústicos

Esta vista tiene como objetivo enumerar y contabilizar los cinco partidos con mayor número de tarjetas.

```
CREATE VIEW partidos_rusticos AS
SELECT
    PARTIDO,
    count(*) as TOTAL
FROM tarjetas
GROUP BY partido
ORDER BY TOTAL DESC
LIMIT 5;
```

	PARTIDO	TOTAL
	15	5
	1	4
	8	4
	9	4
▶	10	4

Segunda vista: Tabla de posiciones

Esta vista propone medir el desempeño de los equipos en el transcurso del torneo. Como la clásica tabla de posiciones realiza conteo de resultados, sumando puntaje obtenido por los equipos ante cada victoria. Consume datos desde la tabla “posiciones”.

```
CREATE VIEW TABLA_POSICIONES AS
SELECT e.nombre, p.puntaje, p.partidos_ganados, p.partidos_perdidos,
p.partidos_jugados, p.empatados
FROM equipos E
INNER JOIN posiciones p
ON e.id_equipo = p.id_equipo
ORDER BY p.puntaje DESC;
```

nombre	puntaje	partidos_ganados	partidos_perdidos	partidos_jugados	empatados
Chaco For Ever	6	2	0	2	0
El Impenetrable FC	6	2	0	2	0
Los Wichi Warriors	6	2	0	2	0
Villa Angela FC	6	2	0	2	0
Atletico Central Benitez	3	1	1	2	0
Charata United	3	1	1	2	0
Club Bancarios	3	1	1	2	0
Futbol Club Barberan	3	1	1	2	0
General Pinedo Stars	3	1	1	2	0
Isla del Cerrito FC	3	1	0	2	1
Los Qom United	3	1	0	2	1
Quitilipi United	3	1	0	2	1
Resistencia FC	3	1	1	2	0
S y Deportivo Guiraldes	3	1	1	2	0
Atletico Central Norte	0	0	1	2	1
Defensores de Vilelas	0	0	2	2	0

Tercer vista: Top jugadore rusticos

Esta vista propone enumerar a los cinco jugadores con mayor número de tarjetas obtenidas, pudiendo así determinar una probabilidad de reincidencia.

```
CREATE VIEW top_rusticos AS
SELECT
    j.nombre,
    count(DISTINCT partido) as total
FROM jugadores j
LEFT JOIN tarjetas t
ON j.national_id = t.national_id
GROUP BY j.nombre
ORDER BY total DESC
limit 5;
```

	nombre	total
▶	Lydia Taylor	3
	Akeem Dominguez	3
	Elliott Williams	2
	Grady Chambers	2
	Dorothy Fleming	2

Tercer vista: Propensos al descenso

En un torneo de todos contra todos se realizan partidos aleatorios y en cada victoria los equipos ganadores obtienen puntaje que al sumarse los posiciona en la tabla.

Así tambien aquellos que menor puntaje obtuvieron al final del campeonato obtienen como resultado un descenso en su categoria, hacia una liga de menor jerarquía.
En esta vista obtenemos los cinco equipos que se posicionan últimos en la tabla de posiciones.

```
CREATE VIEW PROPENSOS_A_DESCENDER AS
SELECT e.nombre as equipo, p.puntaje, p.partidos_ganados, p.partidos_perdidos,
p.partidos_jugados, p.empatados
FROM equipos E
INNER JOIN posiciones p
ON e.id_equipo = p.id_equipo
ORDER BY p.puntaje ASC
LIMIT 5;
```

equipo	puntaje	partidos_ganados	partidos_perdidos	partidos_jugados	empatados
Atletico Central Norte	0	0	1	2	1
Universidad del Nordeste	0	0	1	2	1
Defensores de Vilelas	0	0	2	2	0
Los Tobas FC	0	0	1	2	1
Los Pilaga Stars	0	0	1	2	1

Cuarta vista: Top goleadores

Esta vista propone medir el desempeño de cada jugador en cuanto a sus anotaciones. Contabiliza y promedia la cantidad de goles por partido de manera individual, luego ordena de manera descendente, teniendo en cuenta la cantidad de goles.

```
CREATE VIEW TOP_GOLEADORES AS
SELECT
    j.nombre,
    count(g.id_jugador) as cantidad,
    COUNT(DISTINCT g.id_partido) as partidos_jugados,
    COUNT(g.id_jugador) / COUNT(DISTINCT g.id_partido) AS promedio
FROM jugadores j
RIGHT JOIN goles g
ON j.national_id = g.id_jugador
GROUP BY j.nombre
ORDER BY cantidad DESC
LIMIT 10;
```

	nombre	cantidad	partidos_jugados	promedio
▶	Elliott Tran	5	2	2.5000
	Upton Guthrie	3	2	1.5000
	Julie Blanchard	3	1	3.0000
	Charissa Osborn	3	2	1.5000
	Jessamine Mclaughlin	2	1	2.0000
	Minerva Rocha	2	1	2.0000
	Harriet Lynn	2	1	2.0000
	Madaline Villarreal	2	1	2.0000
	Leslie Ball	2	2	1.0000
	Felicia Hahn	2	1	2.0000

FUNCIONES

Primer función

La siguiente función realiza una busqueda en la tabla jugadores, siendo objetivo retornar si el jugador especificado obtuvo una tarjeta roja. Para realizar la busqueda toma como parámetro el numero de dni.

```
DELIMITER $$
CREATE FUNCTION `RETORNA_JUGADOR_SUSPENDIDO` (DNI INT)
    RETURNS VARCHAR(35)
    READS SQL DATA
BEGIN
    DECLARE NOMBRE VARCHAR(35);
    SET NOMBRE = (SELECT j.nombre
                  FROM jugadores j
                  INNER JOIN tarjetas t
                  ON j.national_id = t.national_id
                  WHERE t.tipo_tarjeta = 'ROJA'
                  AND t.national_id = DNI
                  limit 1);
    RETURN NOMBRE;
END
$$
```

Segunda función

La siguiente función retorna el índice de masa corporal del jugador, teniendo como objetivo establecer la posibilidad del mismo para participar en un partido definido, pudiendo un exceso de peso representar una disminución en el rendimiento.

```
DELIMITER $$
CREATE FUNCTION `RETORNA_IMC` (DNI int)
RETURNS FLOAT
DETERMINISTIC
BEGIN
    DECLARE peso_jugador FLOAT;
    DECLARE estatura_jugador FLOAT;
    DECLARE IMC FLOAT;
    SET peso_jugador = (SELECT peso
                        FROM jugadores
                        WHERE national_id = DNI);
    SET estatura_jugador = (SELECT estatura
                            FROM jugadores
                            WHERE national_id = DNI);

    SET IMC = peso_jugador / POWER(estatura_jugador, 2);
    RETURN IMC;
END
$$
```

STORED PROCEDURE

Primer stored procedure

El primer procedimiento almacenado retorna los registros de la columna jugadores ordenandolos por la columna que el usuario decida.

```
DELIMITER $$
CREATE PROCEDURE `odenar_jugadores` (columna VARCHAR(25), order_by VARCHAR(35))
BEGIN

    IF columna <> '' THEN
        SET @order = CONCAT('ORDER BY ', columna);
    ELSE
        SET @order = '';
    END IF;

IF order_by <> '' THEN
    SET @order_2 = ' DESC';
ELSE
    SET @order_2 = order_by;
END IF;

    SET @clausula = CONCAT('SELECT * FROM jugadores ', @order, @order_2);
    PREPARE runSQL FROM @clausula;
    EXECUTE runSQL;
    DEALLOCATE PREPARE runSQL;
END $$
```

Segundo stored procedure

El siguiente procedimiento almacenado ayuda a la actualización de datos en la tabla jugadores, pudiendo elegir la columna y el registro a modificiar; para esto último toma como primer parámetro el dni para seleccionar al jugador, y en segundo lugar seleccionamos el atributo que deseamos modificar.

```
DELIMITER $$
CREATE PROCEDURE `MODIFICAR_INFORMACION_JUGADORES`(IN columna VARCHAR(25), nuevo_registro VARCHAR(25), DNI INT)
BEGIN

    DECLARE sentence VARCHAR(1000);

    /*encerrar el parametro nuevo_registro entre comillas fue la unica forma de conseguir
    que tome tanto enteros como strings*/
    SET sentence = CONCAT('UPDATE jugadores SET ', columna, ' = ''',
    nuevo_registro, ''' WHERE national_id = ', DNI);

    SET @clausula = sentence;
    PREPARE runSQL FROM @clausula;
    EXECUTE runSQL;
    DEALLOCATE PREPARE runSQL;
END $$
```

TRIGGERS

Primer trigger: auditoria de inserción

Este trigger realiza la auditoria de inserciones en la tabla jugadores.

```
CREATE TABLE cambios_jugador (
    national_id INT,
    create_user VARCHAR(25),
    modify_user VARCHAR(25),
    create_date DATE,
    update_date DATE,
    update_type VARCHAR(20)
);
```

```
DELIMITER $$
CREATE TRIGGER `NUEVO_JUGADOR`
    AFTER INSERT ON jugadores
FOR EACH ROW
BEGIN
    INSERT INTO cambios_jugador (national_id, create_user, modify_user,
    create_date, update_date, update_type)
    VALUES (NEW.national_id, USER(), USER(), CURRENT_TIMESTAMP,
    CURRENT_TIMESTAMP(), 'INSERT');
END
$$
```

Segundo trigger: Actualización de tabla posiciones.

Dada la longitud del mismo resulta contraproducente capturar el código y adjuntarlo en este pdf, para mayor comprensión del mismo revisar el script anexo.

Este trigger realiza la actualización de los valores en la tabala posiciones, lo mismo se logra auditando cada insert realizado en la tabla partidos, midiendo el desempeño de cada equipo participante.

Repositorio

Pulse este boton para teletransportarse a GITHUB