Função

**Função 1 : Serve para calcular a porcentagem de jogadores por Elo**

CREATE OR REPLACE FUNCTION fn\_GetPorcentagemJogadoresPorElo(

elo\_param VARCHAR

)

RETURNS FLOAT

AS $$

DECLARE

total\_jogadores integer;

jogadores\_elo integer;

porcentagem float;

BEGIN

SELECT COUNT(1) INTO total\_jogadores FROM Jogador;

SELECT COUNT(1) INTO jogadores\_elo FROM Jogador WHERE elo = elo\_param;

IF total\_jogadores = 0 THEN

RETURN 0;

ELSE

porcentagem := (jogadores\_elo::FLOAT / total\_jogadores::FLOAT) \* 100;

RETURN porcentagem;

END IF;

END;

$$ LANGUAGE plpgsql;

#Teste da função

select fn\_GetPorcentagemJogadoresPorElo('Desafiante');

select count(1) from jogador;

select count(1) from jogador where elo = 'Desafiante';

**Função 2: Tem o papel de criar uma nova partida trocando os jogadores de lado**

DROP FUNCTION IF EXISTS mudar\_lado\_partida;

CREATE OR REPLACE FUNCTION mudar\_lado\_partida(data\_partida\_input date, sala\_input varchar, numSalaNova char(25))

RETURNS char(10) AS

$$

DECLARE

partida\_data date;

partida\_sala varchar(25);

partida\_id\_azul integer;

partida\_id\_vermelho integer;

anoSessao char(10);

semestreSessao char(10);

atualData DATE;

BEGIN

atualData := CURRENT\_DATE;

SELECT Data\_partida, Sala, IDAzul, IDVermelho, ano, semestre

INTO partida\_data, partida\_sala, partida\_id\_azul, partida\_id\_vermelho, anoSessao, semestreSessao

FROM partida

WHERE partida.data\_partida = data\_partida\_input

AND partida.sala = sala\_input;

INSERT INTO Partida (Data\_partida, Sala, ano, semestre, IDAzul, IDVermelho)

VALUES(atualData, numSalaNova, anoSessao,semestreSessao, partida\_id\_vermelho, partida\_id\_azul);

RETURN numSalaNova;

END;

$$

LANGUAGE PLPGSQL;

#Teste

SELECT mudar\_lado\_partida('2022-12-07', '10', '58' );

select \* from partida

**Função 3: Tem o objetivo de criar uma nova partida e registra a pontuação das equipes.**

DROP FUNCTION IF EXISTS criar\_nova\_partida;

CREATE OR REPLACE FUNCTION criar\_nova\_partida(

data\_partida\_input DATE,

sala\_input VARCHAR,

id\_azul\_input INTEGER,

id\_vermelho\_input INTEGER,

ano\_input CHAR(10),

semestre\_input CHAR(10),

pontuacao\_azul\_input INTEGER,

pontuacao\_vermelho\_input INTEGER

)

RETURNS VOID AS $$

DECLARE

atualData DATE;

BEGIN

atualData := CURRENT\_DATE;

INSERT INTO Partida (Data\_partida, Sala, IDAzul, IDVermelho, Ano, Semestre)

VALUES (data\_partida\_input, sala\_input, id\_azul\_input, id\_vermelho\_input, ano\_input, semestre\_input);

CREATE TABLE IF NOT EXISTS PontuacaoPartida (

Data\_partida DATE,

Sala VARCHAR(25),

PontuacaoAzul INTEGER,

PontuacaoVermelho INTEGER

);

INSERT INTO PontuacaoPartida (Data\_partida, Sala, PontuacaoAzul, PontuacaoVermelho)

VALUES (data\_partida\_input, sala\_input, pontuacao\_azul\_input, pontuacao\_vermelho\_input);

RETURN;

END;

$$ LANGUAGE PLPGSQL;

-- Teste

SELECT criar\_nova\_partida('2022-12-08', '131', 1, 2, '2022', '1º', 4, 9);

SELECT criar\_nova\_partida('2022-12-08', '113', 3, 4, '2022', '1º', 6, 1);

SELECT criar\_nova\_partida('2022-12-09', '555', 5, 6, '2022', '1º', 7, 9);

SELECT \* FROM Pontuacaopartida;

SELECT \* from partida;

**Função 4: Aplica um desconto aos preços de skin conforme o dia, em dias pares temos desconto nos produtos de 975 e em dias ímpares nas skins de 1350 e 520**

CREATE OR REPLACE FUNCTION desconto\_skin()

RETURNS void AS $$

DECLARE

desconto numeric;

BEGIN

IF EXTRACT(DAY FROM CURRENT\_DATE)::integer % 2 = 0 THEN

desconto = 20;

UPDATE skins

SET preço = preço - preço \* (desconto / 100)

WHERE preço = 975;

IF (SELECT COUNT(\*) from skins where preço = 364) > 0 THEN

UPDATE skins

SET preço = 520

WHERE preço = 364;

ELSE

UPDATE skins

SET preço = 975

WHERE preço = 945;

END IF;

ELSE

desconto = 30;

UPDATE skins

SET preço = preço - preço \* (desconto / 100)

WHERE preço = 1350 or preço = 520;

IF (SELECT COUNT(\*) from skins where preço = 780) > 0 THEN

UPDATE skins

SET preço = 975

WHERE preço = 780;

END IF;

END IF;

END;

$$ LANGUAGE plpgsql;

—--- Teste

SELECT \* FROM skinks

SELECT desconto\_skin();

SELECT \* FROM skinks

**Função 5: Serve para calcular o faturamento de uma organização em uma season, levando em consideração os valores ganhos em premiações e gastos em contratações e bônus.**

CREATE OR REPLACE FUNCTION Fat\_org\_season(nome\_org CHAR, endereco\_org char,ano\_s varchar,s\_semestre varchar,porcentagem\_bicho\_jog\_e\_tec int) RETURNS NUMERIC AS $$

DECLARE

Contrato\_porseason float;

premiacoes INT;

BEGIN

SELECT SUM (valor)

INTO contrato\_porseason

from contratacao

WHERE nome\_org = nomeorg AND

endereco\_org = endorg

group by contratacao.nomeorg;

SELECT premiação

into premiacoes

from season as s

where ano\_s = s.ano AND

s\_semestre = s.semestre;

return (premiacoes-(((premiacoes\*porcentagem\_bicho\_jog\_e\_tec)/100)\*6)) - contrato\_porseason;

END;

$$ LANGUAGE plpgsql;

#Teste da função:

SELECT Fat\_org\_season ('Coca-Cola', 'Bangu','2022','2º',5)

= 1370000

Trigger

**Trigger 1: Tem o objetivo de atualizar os elos de todos os jogadores ao iniciar uma nova season.**  
CREATE OR REPLACE FUNCTION atualizar\_elo\_jogador()

RETURNS TRIGGER AS $$

DECLARE

season\_exists INTEGER;

BEGIN

SELECT COUNT(\*) INTO season\_exists

FROM Ranking

WHERE Ano = NEW.Ano AND Semestre = NEW.Semestre;

UPDATE Jogador

SET elo = CASE

WHEN elo = 'Desafiante' THEN 'Diamante'

WHEN elo = 'Grão-Mestre' THEN 'Diamante'

WHEN elo = 'Mestre' THEN 'Diamante'

WHEN elo = 'Diamante' THEN 'Platina'

WHEN elo = 'Platina' THEN 'Ouro'

WHEN elo = 'Ouro' THEN 'Prata'

WHEN elo = 'Prata' THEN 'Bronze'

WHEN elo = 'Bronze' THEN 'Ferro'

ELSE elo

END;

IF season\_exists = 0 THEN

INSERT INTO Ranking (Elo, Ano, Semestre)

VALUES ('Desafiante', NEW.Ano, NEW.Semestre),

('Diamante', NEW.Ano, NEW.Semestre),

('Grão-Mestre', NEW.Ano, NEW.Semestre),

('Mestre', NEW.Ano, NEW.Semestre),

('Platina', NEW.Ano, NEW.Semestre),

('Ouro', NEW.Ano, NEW.Semestre),

('Prata', NEW.Ano, NEW.Semestre),

('Bronze', NEW.Ano, NEW.Semestre);

END IF;

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER mudanca\_temporada

AFTER INSERT ON Season

FOR EACH ROW

EXECUTE FUNCTION atualizar\_elo\_jogador();

– Teste

INSERT INTO Season (Ano, Semestre, Patrocínio, Premiação) VALUES ('2023', '1º', 'VIVARA', 1000000);

SELECT \* FROM jogador;

SELECT \* from ranking;

**Trigger 2: Tem o papel de inserir uma vitoria ao jogador, visto que ela observa a cada nova inserção na tabela PontuacaoPartida.**

DROP TRIGGER IF EXISTS ganhosPartida ON PontuacaoPartida;

DROP FUNCTION IF EXISTS setGanho();

CREATE OR REPLACE FUNCTION setGanho()

RETURNS TRIGGER AS $$

DECLARE

id\_azul INTEGER;

id\_vermelho INTEGER;

data\_partida\_al DATE;

sala\_al VARCHAR(25);

PontuacaoAzul\_al INTEGER;

PontuacaoVermelho\_al INTEGER;

BEGIN

data\_partida\_al := NEW.Data\_partida;

sala\_al := NEW.Sala;

PontuacaoAzul\_al := NEW.PontuacaoAzul;

PontuacaoVermelho\_al := NEW.PontuacaoVermelho;

SELECT IDAzul, IDVermelho INTO id\_azul, id\_vermelho

FROM Partida

WHERE Data\_partida = data\_partida\_al AND Sala = sala\_al;

IF (PontuacaoAzul\_al > PontuacaoVermelho\_al) THEN

UPDATE Jogador

SET partidasGanhas = partidasGanhas + 1

WHERE idEquipe IN (

SELECT id

FROM equipe

WHERE id = id\_azul

);

ELSIF (PontuacaoAzul\_al < PontuacaoVermelho\_al) THEN

UPDATE Jogador

SET partidasGanhas = partidasGanhas + 1

WHERE idEquipe IN (

SELECT id

FROM equipe

WHERE id = id\_vermelho

);

-- ...

ELSE

RETURN NEW;

-- ...

END IF;

RETURN NEW;

END;

$$ LANGUAGE PLPGSQL;

CREATE TRIGGER ganhosPartida

AFTER INSERT OR UPDATE ON PontuacaoPartida

FOR EACH ROW

EXECUTE FUNCTION setGanho();