POSTGRES SQL

```
CREATE DATABASE "Yugioh"
/*CRIAÇÃO DAS TABELAS */
CREATE TABLE CARTA(
        ID BIGSERIAL NOT NULL,
        NOME VARCHAR(45) NOT NULL,
        DETALHES VARCHAR(256),
        ATRIBUTO_ATAQUE INT NOT NULL,
        ATRIBUTO_DEFESA INT NOT NULL,
        IMAGEM_SRC VARCHAR(45),
        PRIMARY KEY(ID));

CREATE TABLE JOGADOR(
        ID BIGSERIAL NOT NULL,
        NOME VARCHAR(45) NOT NULL,
        CPF VARCHAR(11) NOT NULL,
        EMAIL VARCHAR(45),
        PARTIDAS_JOGADAS INT,
        PARTIDAS_GANHAS INT,
        PRIMARY KEY(ID));

 CREATE TABLE PARTIDA(
        ID BIGSERIAL NOT NULL,
        DATA DATE,
        PRIMARY KEY(ID));
        CREATE TABLE PARTIDA_CARTA(
        ID BIGSERIAL NOT NULL,
        CARTA_ID INT,
        PARTIDA_ID INT,
```

```
        PRIMARY KEY(ID),
        CONSTRAINT CARTA_ID
                FOREIGN KEY(CARTA_ID)
                        REFERENCES CARTA,
        CONSTRAINT PARTIDA_ID
                FOREIGN KEY(PARTIDA_ID)
                        REFERENCES PARTIDA);

CREATE TABLE PARTIDA_JOGADOR(
        ID BIGSERIAL NOT NULL,
        JOGADOR_ID INT,
        PARTIDA_ID INT,
        PRIMARY KEY(ID),
        CONSTRAINT JOGADOR_ID
                FOREIGN KEY(JOGADOR_ID)
                        REFERENCES JOGADOR,
        CONSTRAINT PARTIDA_ID
                FOREIGN KEY(PARTIDA_ID)
                        REFERENCES PARTIDA);

CREATE TABLE PARTIDA_CARTA_JOGADOR(
        ID SERIAL NOT NULL,
        ID_JOGADOR INT,
        ID_CARTA INT,
        ID_PARTIDA INT,
        PRIMARY KEY(ID),
        CONSTRAINT ID_JOGADOR
                FOREIGN KEY(ID_JOGADOR)
                        REFERENCES JOGADOR,
```

```
        CONSTRAINT ID_CARTA
                FOREIGN KEY (ID_CARTA)
                        REFERENCES CARTA,
        CONSTRAINT ID_PARTIDA
                FOREIGN KEY (ID_PARTIDA)
                        REFERENCES PARTIDA)


CREATE TABLE RANKING(
        ID SERIAL NOT NULL,
        PORCENTAGEM DOUBLE PRECISION,
        ID_JOGADOR INT NOT NULL,
        PRIMARY KEY (ID),
        CONSTRAINT ID_JOGADOR
                FOREIGN KEY(ID_JOGADOR)
                        REFERENCES JOGADOR)
/* CRIAÇAO DA VIEW */
CREATE VIEW JOGADAS AS ( SELECT CJP.ID_PARTIDA, J.NOME AS JOGADOR, C.NOME AS CARTA FROM
PARTIDA_CARTA_JOGADOR AS CJP
JOIN CARTA AS C ON CJP.ID_CARTA = C.ID
JOIN JOGADOR AS J ON J.ID = CJP.ID_JOGADOR )

 /*CRIAÇAO DOS INDEXS*/
CREATE INDEX NOME_CARTAS ON CARTA(NOME);
CREATE INDEX RANKIN_JOGADOR ON JOGADOR(PARTIDAS_GANHAS);

/*CRIAÇAO DAS FUNCTIONS*/
CREATE FUNCTION JA_EXISTE(NOVO_NOME VARCHAR(45)) RETURNS BOOLEAN AS $$
BEGIN
        IF EXISTS  (SELECT * FROM CARTA WHERE NOME = NOVO_NOME) THEN
```

```
                RETURN TRUE;
        ELSE
                RETURN FALSE;
        END IF;
END
$$ LANGUAGE plpgsql;
--FUNÇÃO USADA PELA VALIDAR CPF
CREATE OR REPLACE FUNCTION VALIDA_DIGITO_CPF(CPF VARCHAR(11), DIGITO INT) RETURNS INT AS $$
DECLARE
 COUNTER INT = 1;
 SOMA INT = 0;
 RESTO INT = 0;
BEGIN
        WHILE COUNTER < 10 LOOP
                SOMA := SOMA + ((11-COUNTER) * SUBSTRING(CPF, DIGITO, 1)::INT);
                COUNTER := COUNTER +1;
                DIGITO := DIGITO +1;
        END LOOP;
        RESTO = (SOMA * 10) % 11;
        IF RESTO = 10 THEN RESTO := 0; END IF;
        RETURN RESTO;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION VALIDA_CPF(CPF VARCHAR(11)) RETURNS BOOLEAN AS $$
BEGIN
        IF (VALIDA_DIGITO_CPF(CPF, 1) = SUBSTRING(CPF, 10, 1)::INT) THEN
                IF(VALIDA_DIGITO_CPF(CPF, 2) = SUBSTRING(CPF,11,1)::INT) THEN
                        RETURN TRUE;
```

```
                ELSE
                        RETURN FALSE;
                END IF;
        ELSE
                RETURN FALSE;
        END IF;
END
$$ LANGUAGE plpgsql;

--CRIAÇÃO  PROCEDURES

CREATE OR REPLACE PROCEDURE SALVAR_CARTA(
NOME VARCHAR(45), DETALHES VARCHAR(256), ATRIBUTO_ATAQUE INT,
ATRIBUTO_DEFESA INT, ID_JOGADOR BIGINT, ID_PARTIDA BIGINT)
LANGUAGE 'plpgsql' AS $$
DECLARE
ID_SALVO BIGINT;
BEGIN
IF NOT (SELECT JA_EXISTE(NOME)) THEN
INSERT INTO CARTA (NOME, DETALHES, ATRIBUTO_ATAQUE, ATRIBUTO_DEFESA)
VALUES (NOME, DETALHES, ATRIBUTO_ATAQUE, ATRIBUTO_DEFESA) RETURNING ID
INTO ID_SALVO;
INSERT INTO CARTA_JOGADOR_PARTIDA (ID_CARTA,ID_JOGADOR, ID_PARTIDA) VALUES (ID_SALVO,
ID_JOGADOR, ID_PARTIDA);
END IF;
IF EXISTS( SELECT * FROM JOGADOR WHERE ID = ID_JOGADOR) THEN COMMIT;
ELSE ROLLBACK;
END IF;
END;
```

```
$$;



 CREATE PROCEDURE ATUALIZA_RANKING() LANGUAGE plpgsql as $$
BEGIN
        DELETE FROM RANKING;
        INSERT INTO RANKING(PORCENTAGEM, ID_JOGADOR) SELECT PARTIDAS_GANHAS::DOUBLE
        PRECISION / PARTIDAS_JOGADAS::DOUBLE PRECISION AS P, J.ID FROM JOGADOR J ORDER BY P
        DESC;
        IF NOT EXISTS (SELECT * FROM RANKING)
                THEN ROLLBACK;
        ELSE
                COMMIT;
        END IF;
END
$$
```

SQL SERVER

```
CREATE DATABASE YUGIOH;
USE YUGIOH;
CREATE TABLE CARTA(
        ID BIGINT IDENTITY PRIMARY KEY NOT NULL,
        NOME VARCHAR(45) NOT NULL,
        DETALHES VARCHAR(256),
        ATRIBUTO_ATAQUE INT NOT NULL,
        ATRIBUTO_DEFESA INT NOT NULL,
        IMAGEM_SRC VARCHAR(45));

 CREATE TABLE JOGADOR(
        ID BIGINT IDENTITY NOT NULL,
        NOME VARCHAR(45) NOT NULL,
        CPF VARCHAR(11) NOT NULL,
        EMAIL VARCHAR(45),
        PARTIDAS_JOGADAS INT,
        PARTIDAS_GANHAS INT,
        PRIMARY KEY(ID));



CREATE TABLE PARTIDA(
```

```
    ID BIGINT IDENTITY NOT NULL,
    DATA DATE,
    PRIMARY KEY(ID));

CREATE TABLE PARTIDA_CARTA_JOGADOR (
    ID BIGINT IDENTITY NOT NULL,
    CARTA_ID BIGINT FOREIGN KEY REFERENCES CARTA(ID),
    PARTIDA_ID BIGINT FOREIGN KEY REFERENCES PARTIDA(ID),
    JOGADOR_ID BIGINT FOREIGN KER REFERENCES JOGADOR(ID),
    PRIMARY KEY(ID));

CREATE TABLE PARTIDA_JOGADOR(
    ID BIGINT IDENTITY PRIMARY KEY NOT NULL,
    JOGADOR_ID BIGINT FOREIGN KEY REFERENCES JOGADOR(ID),
    PARTIDA_ID BIGINT FOREIGN KEY REFERENCES PARTIDA(ID));

CREATE TABLE RANKING(
    ID BIGINT IDENTITY PRIMARY KEY NOT NULL,
    PORCENTAGEM DOUBLE PRECISION,
    ID_JOGADOR BIGINT NOT NULL FOREIGN KEY REFERENCES JOGADOR(ID));

--CRIAÇÃO DA VIEW

CREATE VIEW JOGADAS AS ( SELECT CJP.ID_PARTIDA, J.NOME AS JOGADOR, C.NOME AS CARTA FROM
PARTIDA_CARTA_JOGADOR AS CJP
JOIN CARTA AS C ON CJP.ID_CARTA = C.ID
JOIN JOGADOR AS J ON J.ID = CJP.ID_JOGADOR )
```

```
--CRIAÇÃO DO INDEX

CREATE INDEX NOME_CARTAS ON CARTA(NOME);

CREATE INDEX RANKIN_JOGADOR ON JOGADOR(PARTIDAS_GANHAS);


--CRIAÇÃO DE FUNÇÕES

CREATE FUNCTION JA_EXISTE(@NOVO_NOME VARCHAR(45)) RETURNS BIT

BEGIN

DECLARE @RETORNO BIT

        IF EXISTS  (SELECT * FROM CARTA WHERE NOME = @NOVO_NOME)

SET @RETORNO = 1

ELSE

SET @RETORNO =0

RETURN @RETORNO

END;

CREATE FUNCTION TOTAL_PARTIDAS() RETURNS INT AS

BEGIN

RETURN (SELECT COUNT(PARTIDAS_JOGADAS) FROM JOGADOR);

END

--CRIAÇÃO PROCEDURE

CREATE PROCEDURE ATUALIZA_RANKING AS

BEGIN

DELETE FROM RANKING;

INSERT INTO RANKING(PORCENTAGEM, ID_JOGADOR) SELECT CAST(PARTIDAS_GANHAS AS
FLOAT)/CAST(PARTIDAS_JOGADAS AS FLOAT) AS P, J.ID

FROM JOGADOR J ORDER BY P DESC;

IF NOT EXISTS (SELECT * FROM RANKING)

ROLLBACK

ELSE
```

COMMIT

END