Alunos : Alex Ramos da Silva, Alexandre de Souza Cabral, João Victor de Sousa Guedes e Newton Cardoso da Rocha Neto

Consultas Alex:

-Group by/Having

```
SELECT Treinador.ID, COUNT(PC.Número _na_ Pokédex) AS Total_Pokémon
FROM Treinador
LEFT JOIN Pokémon ON Treinador.ID = Pokémon.ID
LEFT JOIN PC ON Pokémon.Número _na_ Pokédex = PC.Número _na_ Pokédex
GROUP BY Treinador.ID
HAVING COUNT(PC.Número _na_ Pokédex) > 3;
```

-Junção Interna

```
SELECT Treinador.ID, Treinador.Nome, COUNT(Batalha.Desafiante) AS
Total_Batalhas
FROM Treinador
INNER JOIN Batalha ON Treinador.ID = Batalha.Desafiante OR Treinador.ID =
Batalha.Desafiado
GROUP BY Treinador.ID, Treinador.Nome;
```

-Junção Externa

```
SELECT Treinador.ID, Treinador.Nome, Pokémon.Número _na_
Pokédex,Pokémon.Espécie
FROM Treinador
LEFT JOIN Pokémon ON Treinador.ID = Pokémon.ID;
```

-Procedimento SQL embutido em parâmetro

```
CREATE PROCEDURE SelecionarPokemonPorTreinador(
   IN Treinador_ID INT
)
BEGIN
   -- Seleciona os Pokémon do Treinador especificado
   SELECT Número_na_Pokédex, Espécie, Tipo_1, Tipo_2
   FROM Pokémon
   WHERE ID = Treinador_ID;
END;
```

-Chamada:

```
CALL SelecionarPokemonPorTreinador(Treinador.ID);
```

-Função com SQL embutida

```
CREATE FUNCTION CalculaMediaNivelPokemon(TreinadorID INT) RETURNS
FLOAT
BEGIN
   DECLARE Media FLOAT;

   SELECT AVG(Nível)
   INTO Media
```

```
    FROM Pokémon
    WHERE ID = TreinadorID;

    RETURN Media;
END;
```

-Chamada:

```
SELECT CalculaMediaNivelPokemon(1);
```

<u>Consultas Alexandre:</u>

- **subconsulta do tipo linha (**retornam várias colunas, mas apenas uma linha que atendem ao critério)

```
SELECT *
FROM Treinador
WHERE ID IN (SELECT TreinadorID
        FROM Batalha
        WHERE DataDaBatalha > '2023-01-01');
```

Obs: retorna todos os treinadores que participaram de alguma batalha após essa data.

- **subconsulta do tipo escalar** (retornam um único valor)

```
SELECT *
FROM Pokémon
WHERE NúmeroNaPokedex IN (
  SELECT NúmeroNaPokedex
  FROM Captura
  WHERE TreinadorID = 123);
```

Obs: retorna o número na Pokédex dos Pokémons que foram capturados pelo treinador com o ID 123.

- **procedimento SQL embutido em parâmetro**

Meu procedimento é capaz de receber a modalidade como parâmetro, e retornar todos os treinadores que treinam essa modalidade em algum ginásio!

```
CREATE PROCEDURE TreinadoresPorModalidade
  NomeModalidade NVARCHAR(100)
AS
BEGIN
```

```
   DECLARE NVARCHAR(MAX);
      SELECT T.*
      FROM Treinador T
      INNER JOIN Treina Tr ON T.ID = Tr.TreinadorID
      INNER JOIN Ginásio G ON Tr.GinásioID = G.ID
      INNER JOIN Modalidade M ON Tr.ModalidadeID = M.ID
      WHERE M.Nome = ?';

END;
```

chamada do procedimento:

```
EXEC TreinadoresPorModalidade(100);
```

- **função com SQL embutida e gatilho**

minha função é capaz de controlar o fluxo de novos treinadores

```
CREATE FUNCTION fn_Audit_InsertTreinador ()
RETURNS TRIGGER
AS BEGIN
   INSERT INTO AuditoriaTreinador (TreinadorID, Nome, DataInsercao)
   VALUES (NEW.ID, NEW.Nome, NOW());
   RETURN NEW;
END;

CREATE TRIGGER tr_AfterInsertTreinador
AFTER INSERT ON Treinador
FOR EACH ROW
EXECUTE FUNCTION fn_Audit_InsertTreinador();
```

Consultas João Guedes:

Semi Junção:

```
SELECT DISTINCT Treinador.*

FROM Treinador

JOIN Batalha ON Treinador.ID = Batalha.Desafiante OR Treinador.ID =
Batalha.Desafiado;
```

SubConsulta tipo tabela:
SELECT *
FROM Treinador

WHERE ID IN (SELECT Desafiante FROM Batalha);

Alternative:

SELECT Treinador.*

FROM Treinador

JOIN Batalha ON Treinador.ID = Batalha.Desafiante;

Procedimento SQL embutido em parametro

```
CREATE PROCEDURE Atualizar_Lider(IN cidade_param VARCHAR(255), IN novo_lider
VARCHAR(255))
BEGIN
   UPDATE Ginásio
   SET Líder = novo_lider
   WHERE Cidade = cidade_param;
END;
```

Chamada de procedimento

```
CALL Atualizar_Lider('Cidade1', 'Novo Líder');
```

Função :

```
CREATE FUNCTION GetTreinadoresComBatalha()
RETURNS TABLE (
   ID INT
)
AS $$
BEGIN
   RETURN QUERY
   SELECT DISTINCT Treinador.ID
   FROM Treinador
   JOIN Batalha ON Treinador.ID = Batalha.Desafiante OR Treinador.ID =
Batalha.Desafiado;
END;
$$ LANGUAGE plpgsql;

-- Exemplo de uso da função
SELECT * FROM GetTreinadoresComBatalha();
```

Procedure 2:

```
CREATE PROCEDURE GetTreinadoresEnvolvidosEmBatalhas
AS
BEGIN
   SELECT Desafiante AS ID
   FROM Batalha
   UNION
   SELECT Desafiado AS ID
   FROM Batalha;
END;
```

Função 2:

```
CREATE FUNCTION TreinadoresEnvolvidosApenasComoDesafiante(@parametro INT)
RETURNS TABLE AS
RETURN
(
   SELECT Desafiante AS ID
   FROM Batalha
   WHERE Desafiante = @parametro
   EXCEPT
   SELECT Desafiado AS ID
   FROM Batalha
   WHERE Desafiado = @parametro
);
```

Anti-Junção

```
SELECT *
FROM Pokémon p
WHERE NOT EXISTS (
   SELECT 1
   FROM Lendário l
   WHERE p.Numero_na_Pokédex = l.Numero_na_Pokédex
);
```

operação de junção:

```
SELECT Desafiante AS ID
FROM Batalha
EXCEPT
SELECT Desafiado AS ID
FROM Batalha;
```

Consultas Newton:

DDL

Tables

```
CREATE TABLE Ginásio (
    Cidade VARCHAR(255),
    Líder VARCHAR(255),
    PRIMARY KEY (Cidade)
);

CREATE TABLE Insígnia (
    Nome_da_insignia VARCHAR(255),
    Formato VARCHAR(255),
    Cidade VARCHAR(255),
    FOREIGN KEY (Cidade) REFERENCES Ginásio(Cidade),
    PRIMARY KEY (Nome_da_insignia)
);

CREATE TABLE Modalidade (
    Nome_da_Modalidade VARCHAR(255),
    PRIMARY KEY (Nome_da_Modalidade)
);

CREATE TABLE Treinador (
    ID INT,
    PRIMARY KEY (ID)
);

CREATE TABLE Numero_de_Celular (
    ID INT,
    Numero_de_Celular VARCHAR(255),
    FOREIGN KEY (ID) REFERENCES Treinador(ID),
    PRIMARY KEY (ID)
);

CREATE TABLE Prêmio (
    Nome VARCHAR(255),
    PRIMARY KEY (Nome)
);

CREATE TABLE Pokémon (
    Numero_na_Pokédex INT,
    Espécie VARCHAR(255),
    Tipo_1 VARCHAR(255),
```

```
    Tipo_2 VARCHAR(255),
    ID INT,
    FOREIGN KEY (ID) REFERENCES Treinador(ID),
    PRIMARY KEY (Numero_na_Pokédex)
);

CREATE TABLE PC (
    Numero_na_Pokédex INT,
    Data DATE,
    FOREIGN KEY (Numero_na_Pokédex) REFERENCES
Pokémon(Numero_na_Pokédex),
    PRIMARY KEY (Numero_na_Pokédex)
);

CREATE TABLE Lendário (
    Numero_na_Pokédex INT,
    Numero_de_Avistamentos INT,
    FOREIGN KEY (Numero_na_Pokédex) REFERENCES
Pokémon(Numero_na_Pokédex),
    PRIMARY KEY (Numero_na_Pokédex)
);

CREATE TABLE Mítico (
    Numero_na_Pokédex INT,
    Evento VARCHAR(255),
    FOREIGN KEY (Numero_na_Pokédex) REFERENCES
Pokémon(Numero_na_Pokédex),
    PRIMARY KEY (Numero_na_Pokédex)
);

CREATE TABLE Batalha (
    Desafiante INT,
    Desafiado INT,
    Nome VARCHAR(255),
    Data_de_Batalha DATE,
    FOREIGN KEY (Desafiante) REFERENCES Treinador(ID),
    FOREIGN KEY (Desafiado) REFERENCES Treinador(ID),
    FOREIGN KEY (Nome) REFERENCES Prêmio(Nome),
    PRIMARY KEY (Desafiante, Desafiado, Data_de_Batalha)
);

CREATE TABLE Treina (
    Cidade VARCHAR(255),
    ID INT,
```

```
    Nome_da_Modalidade VARCHAR(255),
    FOREIGN KEY (Cidade) REFERENCES Ginásio(Cidade),
    FOREIGN KEY (ID) REFERENCES Treinador(ID),
    FOREIGN KEY (Nome_da_Modalidade) REFERENCES
Modalidade(Nome_da_Modalidade),
    PRIMARY KEY (Cidade, ID, Nome_da_Modalidade)
);
```