Questionário Difícil de SQL

Banco de Dados: Hospital

Estrutura:

1. **Paciente** (ID_Paciente, Nome, Data_Nascimento, Telefone, Email)
2. **Medico** (ID_Medico, Nome, Especialidade, CRM, Salario)
3. **Consulta** (ID_Consulta, ID_Paciente, ID_Medico, Data_Consulta, Diagnostico, Valor)
4. **Exame** (ID_Exame, ID_Consulta, Tipo_Exame, Resultado, Data_Realizacao)
5. **Internacao** (ID_Internacao, ID_Paciente, ID_Medico, Data_Entrada, Data_Saida, Motivo)

Parte A: Construção do Banco (5 questões)

A1. Crie a tabela `Paciente` com restrições apropriadas (PK, auto incremento, check para data de nascimento, unique para email). **A2.** Crie a tabela `Medico` com colunas: ID_Medico, Nome, Especialidade, CRM (único), Salario (decimal). Adicione uma restrição para salário mínimo (3000). **A3.** Crie a tabela `Consulta` com chaves estrangeiras para Paciente e Medico. Adicione uma restrição para que a data da consulta não possa ser no passado (usando GETDATE() como referência). **A4.** Crie a tabela `Exame` com uma chave estrangeira para Consulta. A coluna Tipo_Exame deve ser um dos valores pré-definidos: 'Sangue', 'Urina', 'Raio-X', 'Ultrassom'. **A5.** Crie a tabela `Internacao` com chaves estrangeiras para Paciente e Medico. Adicione uma restrição para que Data_Saida seja maior ou igual a Data_Entrada.

Parte B: Consultas (15 questões)

1. Crie uma CTE para listar todas as consultas do último mês com nome do paciente e nome do médico.

2. Crie uma stored procedure que calcule o total ganho por médico em um período (datas como parâmetros).
3. Crie um trigger que, ao inserir uma internação, verifique se o médico tem CRM ativo (crie uma coluna Ativo na tabela Medico) e se não estiver, retorne um erro.
4. Utilize uma window function (ROW_NUMBER) para rankear os médicos por número de consultas no último ano.
5. Crie uma função que receba o ID de um paciente e retorne a quantidade de consultas nos últimos 6 meses.
6. Crie uma view que mostre o faturamento do hospital por mês (soma de consultas e internações).
7. Escreva uma consulta que use PIVOT para mostrar a quantidade de exames por tipo por mês.
8. Crie um índice clusterizado na tabela Consulta para a coluna Data_Consulta.
9. Escreva uma transação que insira uma consulta e um exame associado, garantindo atomicidade.
10. Utilize MERGE para atualizar a tabela de pacientes com dados de uma tabela temporária.
11. Crie uma constraint de check para garantir que o valor da consulta seja positivo.
12. Escreva uma consulta recursiva para gerar uma série de datas (últimos 30 dias) e contar consultas por dia.
13. Crie uma stored procedure que use cursores para processar cada paciente e atualizar uma coluna de 'Ultima_Consulta'.
14. Escreva uma trigger que audite alterações na tabela Medico (inserções, updates, deleções) em uma tabela de log.
15. Otimize uma consulta que busca consultas com exames, usando índices apropriados e reescrevendo a query se necessário.

Respostas - Questionário Difícil

Parte A

A1:

```sql
CREATE TABLE Paciente (
    ID_Paciente INT PRIMARY KEY IDENTITY(1,1),
    Nome VARCHAR(100) NOT NULL,
    Data_Nascimento DATE NOT NULL CHECK (Data_Nascimento < GETDATE()),
    Telefone VARCHAR(15),
    Email VARCHAR(100) UNIQUE
);
```

A2:

```sql
CREATE TABLE Medico (
    ID_Medico INT PRIMARY KEY IDENTITY(1,1),
    Nome VARCHAR(100) NOT NULL,
    Especialidade VARCHAR(50) NOT NULL,
    CRM VARCHAR(20) UNIQUE NOT NULL,
    Salario DECIMAL(10,2) NOT NULL CHECK (Salario >= 3000),
    Ativo BIT DEFAULT 1
);
```

A3:

```sql
CREATE TABLE Consulta (
    ID_Consulta INT PRIMARY KEY IDENTITY(1,1),
    ID_Paciente INT NOT NULL,
    ID_Medico INT NOT NULL,
    Data_Consulta DATETIME NOT NULL DEFAULT GETDATE(),
    Diagnostico TEXT,
    Valor DECIMAL(10,2) NOT NULL,
    CONSTRAINT FK_Consulta_Paciente FOREIGN KEY (ID_Paciente) REFERENCES
Paciente(ID_Paciente),
    CONSTRAINT FK_Consulta_Medico FOREIGN KEY (ID_Medico) REFERENCES
Medico(ID_Medico),
    CONSTRAINT CHK_Data_Consulta CHECK (Data_Consulta >= GETDATE())
);
```

A4:

```sql
CREATE TABLE Exame (
    ID_Exame INT PRIMARY KEY IDENTITY(1,1),
    ID_Consulta INT NOT NULL,
    Tipo_Exame VARCHAR(20) NOT NULL CHECK (Tipo_Exame IN ('Sangue', 'Urina',
'Raio-X', 'Ultrassom')),
    Resultado TEXT,
    Data_Realizacao DATE NOT NULL DEFAULT GETDATE(),
    CONSTRAINT FK_Exame_Consulta FOREIGN KEY (ID_Consulta) REFERENCES
Consulta(ID_Consulta)
);
```

A5:

```sql
CREATE TABLE Internacao (
    ID_Internacao INT PRIMARY KEY IDENTITY(1,1),
    ID_Paciente INT NOT NULL,
    ID_Medico INT NOT NULL,
    Data_Entrada DATE NOT NULL DEFAULT GETDATE(),
    Data_Saida DATE,
    Motivo TEXT,
    CONSTRAINT FK_Internacao_Paciente FOREIGN KEY (ID_Paciente) REFERENCES
Paciente(ID_Paciente),
    CONSTRAINT FK_Internacao_Medico FOREIGN KEY (ID_Medico) REFERENCES
Medico(ID_Medico),
    CONSTRAINT CHK_Data_Saida CHECK (Data_Saida >= Data_Entrada)
);
```

Parte B

1:

```sql
WITH Consultas_Recentes AS (
    SELECT C.Data_Consulta, P.Nome AS Paciente, M.Nome AS Medico
    FROM Consulta C
    INNER JOIN Paciente P ON C.ID_Paciente = P.ID_Paciente
    INNER JOIN Medico M ON C.ID_Medico = M.ID_Medico
    WHERE C.Data_Consulta >= DATEADD(MONTH, -1, GETDATE())
)
SELECT * FROM Consultas_Recentes;
```

2:

```
CREATE PROCEDURE sp_TotalGanhoMedico
    @DataInicio DATE,
    @DataFim DATE
AS
BEGIN
    SELECT M.Nome, SUM(C.Valor) AS Total
    FROM Medico M
    INNER JOIN Consulta C ON M.ID_Medico = C.ID_Medico
    WHERE C.Data_Consulta BETWEEN @DataInicio AND @DataFim
    GROUP BY M.Nome;
END;
```

3:

```
CREATE TRIGGER tr_VerificaMedicoAtivo
ON Internacao
INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (SELECT 1 FROM inserted I INNER JOIN Medico M ON I.ID_Medico =
M.ID_Medico WHERE M.Ativo = 0)
    BEGIN
        RAISERROR ('Médico não está ativo', 16, 1);
        RETURN;
    END
    INSERT INTO Internacao (ID_Paciente, ID_Medico, Data_Entrada,
Data_Saida, Motivo)
    SELECT ID_Paciente, ID_Medico, Data_Entrada, Data_Saida, Motivo FROM
inserted;
END;
```

4:

```sql
SELECT
    M.Nome,
    COUNT(C.ID_Consulta) AS Num_Consultas,
    ROW_NUMBER() OVER (ORDER BY COUNT(C.ID_Consulta) DESC) AS Ranking
FROM Medico M
LEFT JOIN Consulta C ON M.ID_Medico = C.ID_Medico
WHERE C.Data_Consulta >= DATEADD(YEAR, -1, GETDATE())
GROUP BY M.Nome;
```

5:

```sql
CREATE FUNCTION fn_ConsultasUltimos6Meses (@ID_Paciente INT)
RETURNS INT
AS
BEGIN
    DECLARE @Quantidade INT;
    SELECT @Quantidade = COUNT(*)
    FROM Consulta
    WHERE ID_Paciente = @ID_Paciente
        AND Data_Consulta >= DATEADD(MONTH, -6, GETDATE());
    RETURN @Quantidade;
END;
```

6:

```sql
CREATE VIEW Faturamento_Mensal AS
SELECT
    YEAR(Data_Consulta) AS Ano,
    MONTH(Data_Consulta) AS Mes,
    SUM(Valor) AS Faturamento_Consultas,
    (SELECT SUM(Valor) FROM Internacao WHERE YEAR(Data_Entrada) =
YEAR(Data_Consulta) AND MONTH(Data_Entrada) = MONTH(Data_Consulta)) AS
Faturamento_Internacoes
FROM Consulta
GROUP BY YEAR(Data_Consulta), MONTH(Data_Consulta);
```

7:

```sql
SELECT *
FROM (
    SELECT Tipo_Exame, MONTH(Data_Realizacao) AS Mes
    FROM Exame
) AS SourceTable
PIVOT (
    COUNT(Tipo_Exame)
    FOR Tipo_Exame IN ([Sangue], [Urina], [Raio-X], [Ultrassom])
) AS PivotTable;
```

8:

```sql
CREATE CLUSTERED INDEX IX_Consulta_Data ON Consulta(Data_Consulta);
```

9:

```sql
BEGIN TRANSACTION;
INSERT INTO Consulta (ID_Paciente, ID_Medico, Valor) VALUES (1, 1, 200.00);
DECLARE @ConsultaID INT = SCOPE_IDENTITY();
INSERT INTO Exame (ID_Consulta, Tipo_Exame) VALUES (@ConsultaID, 'Sangue');
COMMIT;
```

10:

```sql
MERGE INTO Paciente AS Target
USING (SELECT * FROM #PacientesTemp) AS Source
ON Target.Email = Source.Email
WHEN MATCHED THEN
    UPDATE SET Target.Telefone = Source.Telefone
WHEN NOT MATCHED THEN
    INSERT (Nome, Data_Nascimento, Telefone, Email)
    VALUES (Source.Nome, Source.Data_Nascimento, Source.Telefone,
Source.Email);
```

11:

```sql
ALTER TABLE Consulta ADD CONSTRAINT CHK_Valor_Positivo CHECK (Valor > 0);
```

12:

```sql
WITH Datas AS (
    SELECT GETDATE() AS Data
    UNION ALL
    SELECT DATEADD(DAY, -1, Data) FROM Datas WHERE Data > DATEADD(DAY, -30,
GETDATE())
)
SELECT D.Data, COUNT(C.ID_Consulta) AS Consultas
FROM Datas D
LEFT JOIN Consulta C ON CONVERT(DATE, C.Data_Consulta) = CONVERT(DATE,
D.Data)
GROUP BY D.Data
ORDER BY D.Data;
```

13:

```
CREATE PROCEDURE sp_AtualizarUltimaConsulta
AS
BEGIN
    DECLARE @ID_Paciente INT, @UltimaConsulta DATE;
    DECLARE cursorPaciente CURSOR FOR
        SELECT ID_Paciente, MAX(Data_Consulta) FROM Consulta GROUP BY
ID_Paciente;

    OPEN cursorPaciente;
    FETCH NEXT FROM cursorPaciente INTO @ID_Paciente, @UltimaConsulta;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        UPDATE Paciente SET Ultima_Consulta = @UltimaConsulta WHERE
ID_Paciente = @ID_Paciente;
        FETCH NEXT FROM cursorPaciente INTO @ID_Paciente, @UltimaConsulta;
    END

    CLOSE cursorPaciente;
    DEALLOCATE cursorPaciente;
END;
```

14:

```sql
CREATE TABLE Log_Medico (
    ID_Log INT PRIMARY KEY IDENTITY(1,1),
    ID_Medico INT,
    Acao VARCHAR(10),
    Data_Hora DATETIME DEFAULT GETDATE()
);

CREATE TRIGGER tr_AuditaMedico
ON Medico
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    INSERT INTO Log_Medico (ID_Medico, Acao)
    SELECT ID_Medico, 'INSERT' FROM inserted
    UNION ALL
    SELECT ID_Medico, 'UPDATE' FROM inserted
    UNION ALL
    SELECT ID_Medico, 'DELETE' FROM deleted;
END;
```

15:

```sql
-- Criar índice para melhorar a busca
CREATE INDEX IX_Consulta_Exame ON Consulta(ID_Consulta) INCLUDE
(ID_Paciente, ID_Medico);
CREATE INDEX IX_Exame_Consulta ON Exame(ID_Consulta);

-- Reescrever a query para usar EXISTS em vez de JOIN se não precisar de
colunas da tabela Exame
SELECT C.*
FROM Consulta C
WHERE EXISTS (SELECT 1 FROM Exame E WHERE E.ID_Consulta = C.ID_Consulta);
```