Questionário Avançado de SQL

Banco de Dados: Financeiro

Estrutura:

1. **Cliente** (ID_Cliente, Nome, Tipo_Pessoa, Documento, Data_Cadastro)
2. **Conta** (ID_Conta, ID_Cliente, Tipo_Conta, Saldo, Data_Abertura)
3. **Transacao** (ID_Transacao, ID_Conta_Origem, ID_Conta_Destino, Valor, Data_Transacao, Tipo_Transacao)
4. **Investimento** (ID_Investimento, ID_Cliente, Tipo_Investimento, Valor_Aplicado, Data_Aplicacao, Data_Resgate)
5. **Cartao_Credito** (ID_Cartao, ID_Cliente, Numero, Validade, Limite, Fatura_Atual)

Parte A: Construção do Banco (5 questões)

A1. Crie a tabela `Cliente` com partição por Tipo_Pessoa (Física/Jurídica). **A2.** Crie a tabela `Conta` com chave estrangeira para Cliente. Adicione uma restrição para que Saldo não possa ser negativo. **A3.** Crie a tabela `Transacao` com chaves estrangeiras para Conta (origem e destino). Adicione uma coluna Status com padrão 'Pendente'. **A4.** Crie a tabela `Investimento` com chave estrangeira para Cliente. Adicione uma restrição para que Data_Resgate seja maior que Data_Aplicacao. **A5.** Crie a tabela `Cartao_Credito` com chave estrangeira para Cliente. Adicione uma coluna calculada para o disponível (Limite - Fatura_Atual).

Parte B: Consultas (15 questões)

1. Crie uma consulta que use APPLY para obter a última transação de cada conta.
2. Utilize funções de janela (LAG/LEAD) para calcular a variação do saldo de uma conta ao longo do tempo.

3. Crie um índice columnstore para a tabela de transações.
4. Escreva uma stored procedure que use TRY…CATCH e transações para transferir valores entre contas.
5. Crie uma função com table-valued parameters para retornar transações dentro de um período.
6. Crie um trigger que audite alterações na tabela Transacao em uma tabela de log.
7. Crie uma view indexada para mostrar o saldo total de cada cliente.
8. Utilize o SQL Server Agent para criar um job que limpe transações antigas (mais de 5 anos).
9. Escreva uma query que use FOR JSON para gerar um extrato em JSON para um cliente.
10. Configure o Always On para alta disponibilidade (descreva os passos).
11. Use o Query Store para identificar queries problemáticas.
12. Crie um plano de manutenção para atualizar estatísticas e rebuild de índices.
13. Implemente a criptografia de dados sensíveis na tabela Cliente (coluna Documento).
14. Use o PolyBase para consultar dados externos em Hadoop ou Azure Blob Storage.
15. Configure a segurança em nível de linha (Row-Level Security) para que clientes vejam apenas suas próprias transações.

Respostas - Questionário Avançado

Parte A

A1:

```sql
-- Criar função de partição
CREATE PARTITION FUNCTION pf_TipoPessoa (CHAR(1))
AS RANGE LEFT FOR VALUES ('F', 'J');

-- Criar esquema de partição
CREATE PARTITION SCHEME ps_TipoPessoa
AS PARTITION pf_TipoPessoa
ALL TO ([PRIMARY]);

-- Criar tabela particionada
CREATE TABLE Cliente (
    ID_Cliente INT IDENTITY(1,1),
    Nome VARCHAR(100) NOT NULL,
    Tipo_Pessoa CHAR(1) NOT NULL CHECK (Tipo_Pessoa IN ('F', 'J')),
    Documento VARCHAR(20) NOT NULL UNIQUE,
    Data_Cadastro DATE DEFAULT GETDATE(),
    PRIMARY KEY (ID_Cliente, Tipo_Pessoa)
) ON ps_TipoPessoa(Tipo_Pessoa);
```

A2:

```sql
CREATE TABLE Conta (
    ID_Conta INT PRIMARY KEY IDENTITY(1,1),
    ID_Cliente INT NOT NULL,
    Tipo_Conta VARCHAR(20) NOT NULL CHECK (Tipo_Conta IN ('Corrente',
'Poupança', 'Investimento')),
    Saldo DECIMAL(15,2) NOT NULL DEFAULT 0 CHECK (Saldo >= 0),
    Data_Abertura DATE NOT NULL DEFAULT GETDATE(),
    CONSTRAINT FK_Conta_Cliente FOREIGN KEY (ID_Cliente) REFERENCES
Cliente(ID_Cliente)
);
```

A3:

```
CREATE TABLE Transacao (
    ID_Transacao INT PRIMARY KEY IDENTITY(1,1),
    ID_Conta_Origem INT NOT NULL,
    ID_Conta_Destino INT,
    Valor DECIMAL(15,2) NOT NULL CHECK (Valor > 0),
    Data_Transacao DATETIME NOT NULL DEFAULT GETDATE(),
    Tipo_Transacao VARCHAR(20) NOT NULL CHECK (Tipo_Transacao IN
('Depósito', 'Saque', 'Transferência', 'Pagamento')),
    Status VARCHAR(20) DEFAULT 'Pendente' CHECK (Status IN ('Pendente',
'Concluída', 'Cancelada')),
    CONSTRAINT FK_Transacao_Origem FOREIGN KEY (ID_Conta_Origem) REFERENCES
Conta(ID_Conta),
    CONSTRAINT FK_Transacao_Destino FOREIGN KEY (ID_Conta_Destino)
REFERENCES Conta(ID_Conta)
);
```

A4:

```
CREATE TABLE Investimento (
    ID_Investimento INT PRIMARY KEY IDENTITY(1,1),
    ID_Cliente INT NOT NULL,
    Tipo_Investimento VARCHAR(30) NOT NULL CHECK (Tipo_Investimento IN
('CDB', 'Ações', 'Fundo Imobiliário', 'Tesouro Direto')),
    Valor_Aplicado DECIMAL(15,2) NOT NULL CHECK (Valor_Aplicado > 0),
    Data_Aplicacao DATE NOT NULL DEFAULT GETDATE(),
    Data_Resgate DATE,
    CONSTRAINT FK_Investimento_Cliente FOREIGN KEY (ID_Cliente) REFERENCES
Cliente(ID_Cliente),
    CONSTRAINT CHK_Data_Resgate CHECK (Data_Resgate > Data_Aplicacao)
);
```

A5:

```
CREATE TABLE Cartao_Credito (
    ID_Cartao INT PRIMARY KEY IDENTITY(1,1),
    ID_Cliente INT NOT NULL,
    Numero VARCHAR(20) NOT NULL UNIQUE,
    Validade DATE NOT NULL,
    Limite DECIMAL(10,2) NOT NULL,
    Fatura_Atual DECIMAL(10,2) DEFAULT 0,
    Disponivel AS (Limite - Fatura_Atual),
    CONSTRAINT FK_Cartao_Cliente FOREIGN KEY (ID_Cliente) REFERENCES
Cliente(ID_Cliente)
);
```

Parte B

1:

```
SELECT C.*, UT.*
FROM Conta C
CROSS APPLY (
    SELECT TOP 1 *
    FROM Transacao T
    WHERE T.ID_Conta_Origem = C.ID_Conta OR T.ID_Conta_Destino = C.ID_Conta
    ORDER BY Data_Transacao DESC
) UT;
```

2:

```
SELECT
    ID_Conta,
    Data_Transacao,
    Valor,
    LAG(Saldo, 1, Saldo) OVER (PARTITION BY ID_Conta ORDER BY
Data_Transacao) AS Saldo_Anterior,
    Saldo - LAG(Saldo, 1, Saldo) OVER (PARTITION BY ID_Conta ORDER BY
Data_Transacao) AS Variacao
FROM (
    SELECT
        ID_Conta_Origem AS ID_Conta,
        Data_Transacao,
        Valor,
        SUM(CASE WHEN ID_Conta_Origem = ID_Conta THEN -Valor ELSE Valor END)
            OVER (PARTITION BY ID_Conta_Origem ORDER BY Data_Transacao) AS
Saldo
    FROM Transacao
) AS SaldoCalculado;
```

3:

```
CREATE COLUMNSTORE INDEX IX_Transacao_Columnstore ON Transacao(ID_Transacao,
ID_Conta_Origem, ID_Conta_Destino, Valor, Data_Transacao);
```

4:

```
CREATE PROCEDURE sp_Transferencia
    @Origem INT,
    @Destino INT,
    @Valor DECIMAL(15,2)
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        UPDATE Conta SET Saldo = Saldo - @Valor WHERE ID_Conta = @Origem;
        IF @@ROWCOUNT = 0 THROW 50001, 'Conta origem não encontrada', 1;

        UPDATE Conta SET Saldo = Saldo + @Valor WHERE ID_Conta = @Destino;
        IF @@ROWCOUNT = 0 THROW 50002, 'Conta destino não encontrada', 1;

        INSERT INTO Transacao (ID_Conta_Origem, ID_Conta_Destino, Valor,
Tipo_Transacao, Status)
        VALUES (@Origem, @Destino, @Valor, 'Transferência', 'Concluída');

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        IF @@TRANCOUNT > 0 ROLLBACK TRANSACTION;
        THROW;
    END CATCH
END;
```

5:

```
CREATE TYPE Periodo AS TABLE (
    DataInicio DATE,
    DataFim DATE
);

CREATE FUNCTION fn_TransacoesPorPeriodo (@Periodo Periodo READONLY)
RETURNS TABLE
AS
RETURN (
    SELECT T.*
    FROM Transacao T
    INNER JOIN @Periodo P ON T.Data_Transacao BETWEEN P.DataInicio AND
P.DataFim
);
```

6:

```
CREATE TABLE Audit_Transacao (
    AuditID INT PRIMARY KEY IDENTITY(1,1),
    ID_Transacao INT,
    Acao VARCHAR(10),
    DataHora DATETIME DEFAULT GETDATE(),
    Usuario VARCHAR(100) DEFAULT SUSER_NAME()
);

CREATE TRIGGER tr_Audit_Transacao
ON Transacao
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    INSERT INTO Audit_Transacao (ID_Transacao, Acao)
    SELECT ID_Transacao, 'INSERT' FROM inserted
    UNION ALL
    SELECT ID_Transacao, 'UPDATE' FROM inserted
    UNION ALL
    SELECT ID_Transacao, 'DELETE' FROM deleted;
END;
```

7:

```sql
CREATE VIEW vw_SaldoCliente
WITH SCHEMABINDING
AS
SELECT
    C.ID_Cliente,
    C.Nome,
    SUM(CO.Saldo) AS Saldo_Total
FROM dbo.Cliente C
INNER JOIN dbo.Conta CO ON C.ID_Cliente = CO.ID_Cliente
GROUP BY C.ID_Cliente, C.Nome;

CREATE UNIQUE CLUSTERED INDEX IX_vw_SaldoCliente ON
vw_SaldoCliente(ID_Cliente);
```

8:

```sql
-- Criar procedure para limpeza
CREATE PROCEDURE sp_LimparTransacoesAntigas
AS
BEGIN
    DELETE FROM Transacao WHERE Data_Transacao < DATEADD(YEAR, -5,
GETDATE());
END;

-- Configurar job no SQL Server Agent (passos descritos)
-- 1. Criar novo job
-- 2. Adicionar passo com a execução da procedure
-- 3. Agendar para rodar mensalmente
```

9:

```sql
SELECT
    C.Nome,
    C.Documento,
    (
        SELECT
            T.Data_Transacao,
            T.Valor,
            T.Tipo_Transacao,
            CO.Numero AS Conta_Origem,
            CD.Numero AS Conta_Destino
        FROM Transacao T
        LEFT JOIN Conta CO ON T.ID_Conta_Origem = CO.ID_Conta
        LEFT JOIN Conta CD ON T.ID_Conta_Destino = CD.ID_Conta
        WHERE CO.ID_Cliente = C.ID_Cliente OR CD.ID_Cliente = C.ID_Cliente
        FOR JSON PATH
    ) AS Extrato
FROM Cliente C
WHERE C.ID_Cliente = 1
FOR JSON PATH, ROOT('ExtratoCliente');
```

10: (Descrição dos passos)

```
1. Configurar o Windows Server Failover Clustering (WSFC)
2. Instalar o SQL Server em cada nó com a mesma versão
3. Configurar os nós no SQL Server Configuration Manager
4. Criar um listener para o grupo de disponibilidade
5. Configurar o banco de dados para usar o Always On
6. Adicionar réplicas e configurar o modo de failover
```

11:

```sql
-- Habilitar Query Store
ALTER DATABASE [SeuBanco] SET QUERY_STORE = ON;

-- Consultar queries problemáticas
SELECT
    q.query_id,
    qt.query_sql_text,
    rs.avg_duration,
    rs.avg_cpu_time,
    rs.avg_logical_io_reads
FROM sys.query_store_query q
JOIN sys.query_store_query_text qt ON q.query_text_id = qt.query_text_id
JOIN sys.query_store_plan p ON q.query_id = p.query_id
JOIN sys.query_store_runtime_stats rs ON p.plan_id = rs.plan_id
ORDER BY rs.avg_duration DESC;
```

12:

```sql
-- Criar plano de manutenção via T-SQL
USE [master];
GO

EXEC sp_add_job @job_name = 'Manutencao_Indices_Estatisticas';
EXEC sp_add_jobstep @job_name = 'Manutencao_Indices_Estatisticas',
    @step_name = 'Atualizar_Estatisticas',
    @command = 'EXEC sp_updatestats';
EXEC sp_add_jobstep @job_name = 'Manutencao_Indices_Estatisticas',
    @step_name = 'Rebuild_Indices',
    @command = 'ALTER INDEX ALL ON SuaTabela REBUILD';
EXEC sp_add_schedule @schedule_name = 'Diariamente_MeiaNoite',
    @freq_type = 4, -- Diariamente
    @active_start_time = 000000; -- Meia-noite
EXEC sp_attach_schedule @job_name = 'Manutencao_Indices_Estatisticas',
    @schedule_name = 'Diariamente_MeiaNoite';
```

13:

```
-- Criar chave mestra
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'SenhaForte123!';

-- Criar certificado
CREATE CERTIFICATE CertificadoCliente WITH SUBJECT = 'Criptografia Documento
Cliente';

-- Criar chave simétrica
CREATE SYMMETRIC KEY ChaveDocumento WITH ALGORITHM = AES_256
ENCRYPTION BY CERTIFICATE CertificadoCliente;

-- Adicionar coluna para documento criptografado
ALTER TABLE Cliente ADD Documento_Criptografado VARBINARY(MAX);

-- Criptografar dados existentes
OPEN SYMMETRIC KEY ChaveDocumento DECRYPTION BY CERTIFICATE
CertificadoCliente;
UPDATE Cliente SET Documento_Criptografado =
ENCRYPTBYKEY(KEY_GUID('ChaveDocumento'), Documento);
CLOSE SYMMETRIC KEY ChaveDocumento;
```

14:

```
-- Configurar PolyBase
EXEC sp_configure @configname = 'polybase enabled', @configvalue = 1;
RECONFIGURE;

-- Criar credencial externa
CREATE DATABASE SCOPED CREDENTIAL CredencialAzure
WITH IDENTITY = 'SHARED ACCESS SIGNATURE',
SECRET = 'sv=2020-02-10...';

-- Criar fonte de dados externa
CREATE EXTERNAL DATA SOURCE AzureBlobStorage
WITH (
    TYPE = HADOOP,
    LOCATION = 'wasbs://container@storageaccount.blob.core.windows.net',
    CREDENTIAL = CredencialAzure
);

-- Criar tabela externa
CREATE EXTERNAL TABLE TransacoesExternas (
    ID INT,
    Valor DECIMAL(15,2),
    Data DATE
)
WITH (
    LOCATION = '/dados/transacoes.csv',
    DATA_SOURCE = AzureBlobStorage,
    FILE_FORMAT = TextFileFormat
);
```

15:

```sql
-- Criar função de segurança
CREATE FUNCTION fn_SegurancaCliente(@ClienteID AS INT)
RETURNS TABLE
WITH SCHEMABINDING
AS
RETURN SELECT 1 AS fn_SegurancaCliente_result
WHERE @ClienteID = CAST(SESSION_CONTEXT(N'ClienteID') AS INT);

-- Criar política de segurança
CREATE SECURITY POLICY PoliticaCliente
ADD FILTER PREDICATE dbo.fn_SegurancaCliente(ID_Cliente) ON dbo.Transacao;

-- Definir contexto de sessão
EXEC sp_set_session_context @key = N'ClienteID', @value = 1;
```