```sql
IF NOT EXISTS(SELECT * FROM sys.databases WHERE name = 'bd08102024')
    CREATE DATABASE bd08102024;
GO
USE bd08102024;

IF NOT EXISTS(SELECT * FROM sys.tables WHERE name = 'Clientes')
    CREATE TABLE Clientes (
        IdCliente INT PRIMARY KEY IDENTITY(1,1),
        Nome VARCHAR(100),
        Email VARCHAR(100)
    );

/*SavePoints no T-SQL*/

BEGIN TRANSACTION tr1;

PRINT(@@TRANCOUNT);

SELECT * FROM Clientes;

INSERT INTO Clientes VALUES (CONVERT(VARCHAR(100), NEWID()), CONCAT
  ('email_teste_',ROUND(RAND()*10,0),'@hotmail.com'));

SELECT * FROM Clientes;

    SAVE TRANSACTION tr2;

    PRINT(@@TRANCOUNT);


    INSERT INTO Clientes VALUES (CONVERT(VARCHAR(100), NEWID()), CONCAT
      ('email_teste_',ROUND(RAND()*10,0),'@gmail.com'));

    SELECT * FROM Clientes;

    COMMIT TRANSACTION tr2;

    ROLLBACK TRANSACTION tr1;

    SELECT * FROM Clientes;

/*Ex. 1: Crie um procedimento que adiciona um número @n de clientes. Na sequência:
i. Confira o tempo em segundos que leva para adicionar 10000 registros aleatórios.
ii. Envolva o código interno ao procedimento em uma transação confirmada e
  recalcule o tempo necessário.*/

CREATE OR ALTER PROCEDURE add_cliente
@n INT
AS
BEGIN
    BEGIN TRANSACTION;

    DECLARE @i INT = 1;
    WHILE @i <= @n
    BEGIN
        INSERT INTO Clientes
```

```
        VALUES (CONVERT(VARCHAR(100), NEWID()), CONCAT('email_teste_',ROUND(RAND()
          *10,0),'@gmail.com'));

        SET @i = @i + 1;
    END;

    COMMIT TRANSACTION;
END;

DECLARE @start DATETIME = GETDATE();
EXEC add_cliente 10000;
DECLARE @end FLOAT = DATEDIFF(SECOND,GETDATE(),@start);

PRINT(CONCAT('Tempo em segundos: ', ABS(@end)));

/*
Ex. 2: Crie uma tabela tb_contas com id INT PRIMARY KEY IDENTITY e saldo DECIMAL
  (10,2). Crie
um procedimento que transfira um valor do saldo de um cliente para o outro de forma
   a garantir
que a transferência seja realizada de maneira completa e segura. Isto é, que um
  cliente
vai receber um valor no saldo enquanto que o saldo do outro será reduzido nesta
  mesma
quantidade.
*/

CREATE TABLE tb_contas (
    id INT PRIMARY KEY IDENTITY(1,1),
    saldo DECIMAL(10, 2)
);

-- Inserir dados de exemplo
INSERT INTO tb_contas (id, saldo) VALUES (1, 500.00), (2, 300.00);

-- Procedimento para transferir saldo
CREATE OR ALTER PROCEDURE sp_transferir_saldo
    @id_remetente INT,
    @id_destinatario INT,
    @valor DECIMAL(10, 2)
AS
BEGIN
    BEGIN TRANSACTION; -- Iniciar a transação

    BEGIN TRY
        -- Verificar se o remetente tem saldo suficiente
        DECLARE @saldo_remetente DECIMAL(10, 2);
        SELECT @saldo_remetente = saldo FROM tb_contas WHERE id = @id_remetente;

        IF @saldo_remetente < @valor
        BEGIN
            RAISERROR('Saldo insuficiente na conta remetente.', 16, 1);
            ROLLBACK TRANSACTION; -- Reverter a transação se o saldo for
              insuficiente
            RETURN;
        END
```

```
        -- Debitar o valor da conta remetente
        UPDATE tb_contas
        SET saldo = saldo - @valor
        WHERE id = @id_remetente;

        -- Creditar o valor na conta destinatária
        UPDATE tb_contas
        SET saldo = saldo + @valor
        WHERE id = @id_destinatario;

        COMMIT TRANSACTION; -- Confirmar a transação
        PRINT 'Transferência realizada com sucesso.';
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION; -- Reverter a transação em caso de erro
        PRINT 'Erro na transferência: ' + ERROR_MESSAGE();
    END CATCH;
END;

-- Executar a transferência
EXEC sp_transferir_saldo @id_remetente = 3, @id_destinatario = 2, @valor = 200.00;

-- Verificar os saldos
SELECT * FROM tb_contas;


/*
Ex. 3: Crie uma tabela Voos com os campos id INT PRIMARY KEY IDENTITY, e
  assentos_disponiveis INT.
Crie também uma tabela Passageiros com os campos id INT PRIMARI KEY IDENTITY, nome
  VARCHAR(MAX) e voo_id INT, sendo
este último uma referência ao id da tabela Voos. Escreva um procedimento que faça a
   reserva de um assento em um voo
específico para um cliente específico de maneira segura.
*/

CREATE TABLE voos (
    id INT PRIMARY KEY,
    assentos_disponiveis INT
);

CREATE TABLE passageiros (
    id INT PRIMARY KEY IDENTITY(1,1),
    nome VARCHAR(100),
    voo_id INT,
    FOREIGN KEY (voo_id) REFERENCES voos(id)
);

-- Inserir dados de exemplo
INSERT INTO voos (id, assentos_disponiveis) VALUES (1, 10);

-- Procedimento para reservar um assento
CREATE PROCEDURE sp_reservar_assento
    @voo_id INT,
    @passageiro_nome VARCHAR(100)
```

```
AS
BEGIN
    BEGIN TRANSACTION; -- Iniciar a transação

    BEGIN TRY
        -- Verificar assentos disponíveis
        DECLARE @assentos_disponiveis INT;
        SELECT @assentos_disponiveis = assentos_disponiveis FROM voos WHERE id =
          @voo_id;

        IF @assentos_disponiveis <= 0
        BEGIN
            RAISERROR('Não há assentos disponíveis.', 16, 1);
            ROLLBACK TRANSACTION; -- Reverter se não houver assentos
            RETURN;
        END

        -- Atualizar assentos disponíveis
        UPDATE voos
        SET assentos_disponiveis = assentos_disponiveis - 1
        WHERE id = @voo_id;

        -- Adicionar o passageiro
        INSERT INTO passageiros (nome, voo_id)
        VALUES (@passageiro_nome, @voo_id);

        COMMIT TRANSACTION; -- Confirmar a transação
        PRINT 'Reserva realizada com sucesso.';
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION; -- Reverter a transação em caso de erro
        PRINT 'Erro na reserva: ' + ERROR_MESSAGE();
    END CATCH;
END;

-- Executar a reserva
EXEC sp_reservar_assento @voo_id = 1, @passageiro_nome = 'João Silva';

-- Verificar os resultados
SELECT * FROM voos;
SELECT * FROM passageiros;


/*
Ex. 4: Você possui uma tabela chamada tb_contas que armazena informações de contas
  bancárias,
incluindo o saldo de cada conta. O objetivo é transferir uma quantia de uma conta
  para outra,
utilizando savepoints para garantir a integridade dos dados. Se ocorrer um erro
  durante o processo,
você deve reverter apenas parte das operações realizadas
*/
DROP TABLE tb_contas;

CREATE TABLE tb_contas (
    id INT PRIMARY KEY IDENTITY(1,1),
```

```
    nome VARCHAR(100),
    saldo DECIMAL(10,2)
);

INSERT INTO tb_contas (nome, saldo) VALUES ('Conta A', 1000.00);
INSERT INTO tb_contas (nome, saldo) VALUES ('Conta B', 500.00);
INSERT INTO tb_contas (nome, saldo) VALUES ('Conta C', 300.00);


BEGIN TRY
    BEGIN TRANSACTION;

    -- Transferindo 200 da Conta A para a Conta B
    DECLARE @quantia DECIMAL(10,2) = 200.00;

    UPDATE tb_contas
    SET saldo = saldo - @quantia
    WHERE nome = 'Conta A';

    -- Savepoint após a primeira transferência
    SAVE TRANSACTION TransferB;

    UPDATE tb_contas
    SET saldo = saldo + @quantia
    WHERE nome = 'Conta B';

    -- Transferindo 100 da Conta A para a Conta C
    SET @quantia = 100.00;

    UPDATE tb_contas
    SET saldo = saldo - @quantia
    WHERE nome = 'Conta A';

    -- Aqui pode ocorrer um erro se o saldo ficar negativo
    UPDATE tb_contas
    SET saldo = saldo + @quantia
    WHERE nome = 'Conta C';

    COMMIT TRANSACTION;

END TRY
BEGIN CATCH
    -- Se ocorrer um erro, reverter para o savepoint
    IF @@TRANCOUNT > 0
    BEGIN
        ROLLBACK TRANSACTION TransferB; -- Reverte apenas a transferência para
          Conta B
        -- Aqui você pode optar por fazer um rollback total se preferir
        -- ROLLBACK TRANSACTION;
    END

    -- Exibir mensagem de erro
    PRINT 'Erro: ' + ERROR_MESSAGE();
END CATCH;
```