FACULDADE DE TECNOLOGIA SENAC RIO	
Curso: Análise e Desenvolvimento de Sistemas	**Semestre letivo: 2024.1**
Unidade Curricular: Banco de Dados II	**Módulo:** 3
Professor: Roberto Harkovsky	**Data:**
Competências a serem avaliadas: ▪ Administrar sistema gerenciador de banco de dados relacional em sistemas corporativos. ▪ Analisar uso de banco de dados não relacionais em sistemas corporativos.	**Indicadores de Competência:** ▪ Elabora consultas SQL conforme requisitos de desempenho do sistema. ▪ Avalia e implementa corretamente o uso de transações e de objetos (Indices, Views, Funcion, Stored Procedures e Triggers) no contexto de um SGBD.
Aluno: Erick Calazães da Silva	**Conceito:**

Avaliação prática – Construção de *Stored Procedures* e *functions e triggers* utilizando comandos SQL

Ambulatorio (nroa, andar, capacidade)
Medicos (codm, CPF, nome, idade, cidade, especialidade, *nroa*)
Pacientes (codp, CPF, nome, sexo, idade, cidade, doença)
Consultas (*codm*, *codp*, data, hora)
Funcionarios (codf, CPF, nome, idade, cidade, salario)
GrpRisco(Data, Paciente, Doença)

1) Crie um procedimento armazenado que escreva a frase “SENAC RIO”

```
create procedure p_senac
as
    begin
        print 'SENAC RIO'
    end

exec p_senac
```

2) Crie um procedimento armazenado que liste os pacientes com sarampo

```
create procedure p_pac_sarampo (@p1 as char (20))
as
    select Nome, Sexo, Idade, Doença
    from Pacientes
    where doença = @p1

exec p_pac_sarampo 'Sarampo'
```

3) Crie um procedimento armazenado que selecione e liste os pacientes com determinada doença (passada como parâmetro)

```
create procedure p_pac_doenca (@p1 as char (20))
as
    select Nome, Sexo, Idade, Doença
    from Pacientes
    where doença = @p1

exec p_pac_doenca 'Diabetes'
```

4) Listar nome e dados de consultas de determinado paciente (parâmetro = codp)

```
create procedure p_pac_consulta (@p1 as int)
as
    select nome, p.codp codm, data, hora
    from pacientes p inner join consultas c on p.codp = c.codp
    where p.codp = @p1

exec p_pac_consulta 1070
```

5) Crie uma função que calcule o cubo de um numero X

```
create function calc_cubo(@x float)
        returns float
        begin
            return ( @x * @x * @x );
        end

select dbo.calc_cubo(2) as Cubo
```

6) Crie uma função que calcule a raiz de uma equação do primeiro grau do tipo Ax+B=0, dados os coeficientes A e B (dica x=-B/A)

```
create function calc_raiz(@a float, @b float)
        returns float
        begin
            declare @x float;
            set @x = (-@b / @a);
            return @x;
        end

select dbo.calc_raiz(1,10) as Raiz
```

7) Crie uma função chamada ConsultasApos que retorne as consultas com data posterior a um parâmetro passado (teste com ‘01/10/2020’)

```
create function ConsultasApos (@dt datetime)
returns table
    as
        return ( select * from consultas where data >= @dt );

select * from ConsultasApos ('01/10/2020')
```

8) Criar uma trigger que se cadastrar um novo paciente verifique se o mesmo é do grupo de risco (‘Diabetes’, ‘Hipertensão’, Zica’) e caso seja, o registre na tabela GrpRisco (dica: utilize a função do SQL GETDATE() para obter a data/hora do registro para o campo ‘data’)

```
create trigger check_risco
on pacientes
after insert
as
begin
    insert into GrpRisco(codp, nome, data, doenca)
    select codp, nome, getdate(), doença
    from inserted
    where doença IN ('Diabetes', 'Hipertensão', 'Zica')
end
```