Documentação do Banco de Dados

Academia Wakanda

OBJETIVO DO PROJETO

Com as pessoas se preocupando mais com sua saúde, estética e condicionamento físico, o mercado fitness vem crescendo bastante nos últimos anos. E com isso o negócio de academias sentiu a necessidade de um sistema de gerenciamento cujo objetivo é otimizar o processo de cadastro e gerenciamento de professores, alunos e turmas em sistemas de Academias de Ginástica. Dessa forma tornando rápida, por exemplo, a busca dos dados referentes aos planos de treinamento dos alunos. Substituindo assim as fichas de papel que atrasam as buscas, e ocupam desnecessariamente os que estão buscando tais informações.

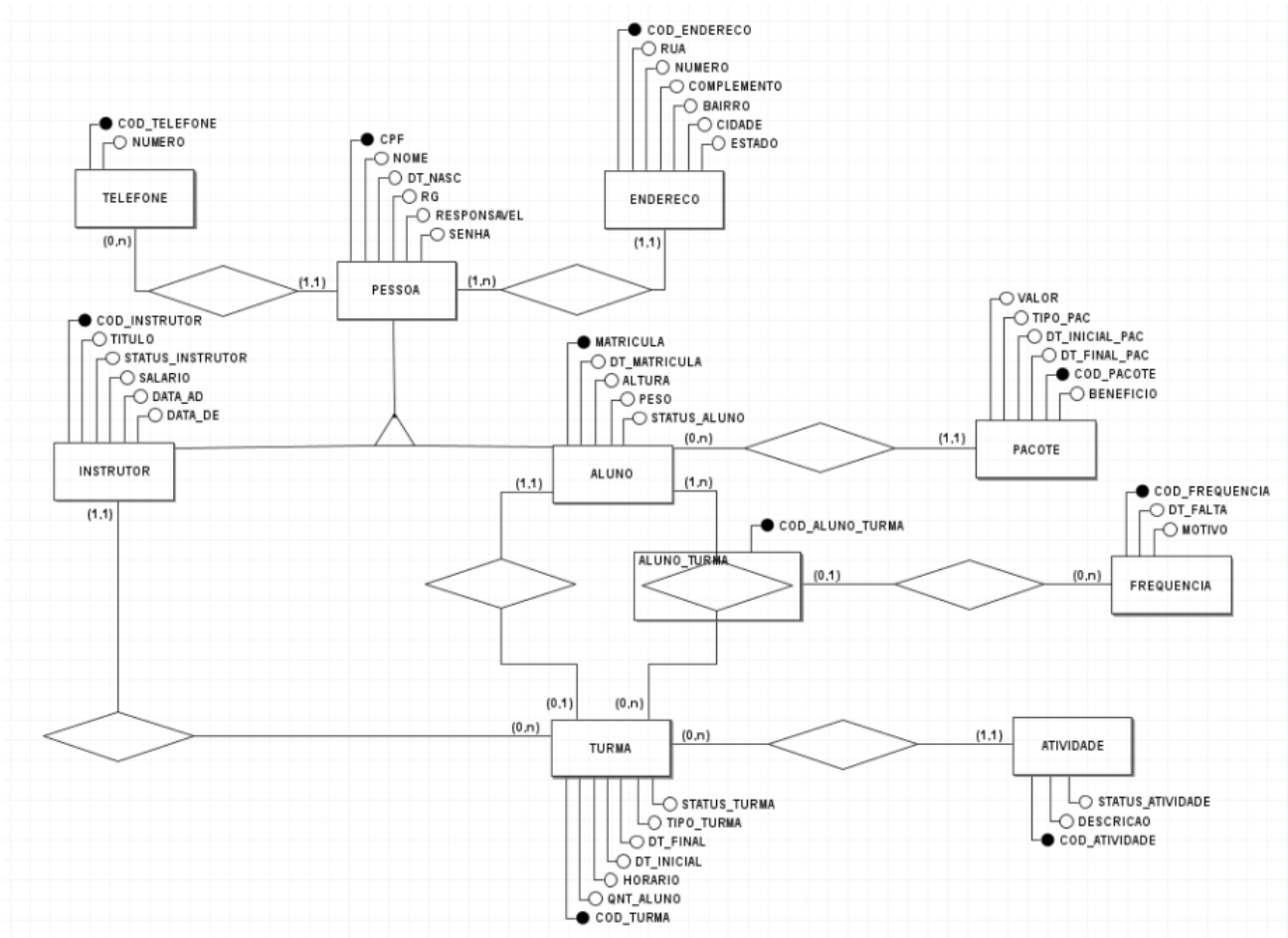
MOTIVAÇÃO / NECESSIDADE DO PROJETO

O sistema foi desenvolvido baseado em pesquisas de mercado que mostram a dificuldade que muitas Academias de Ginástica têm em organizar o vasto número de dados que necessitam para funcionar da melhor forma. Um dos problemas comumente relatados pelos proprietários refere-se a dificuldade em se obter de forma ágil dados essenciais à realização das atividades diárias destas empresas. Como diferenciais, o nosso sistema conta com três pacotes distintos de adesão. Sendo eles, o Pacote Bronze, Prata e o Ouro.

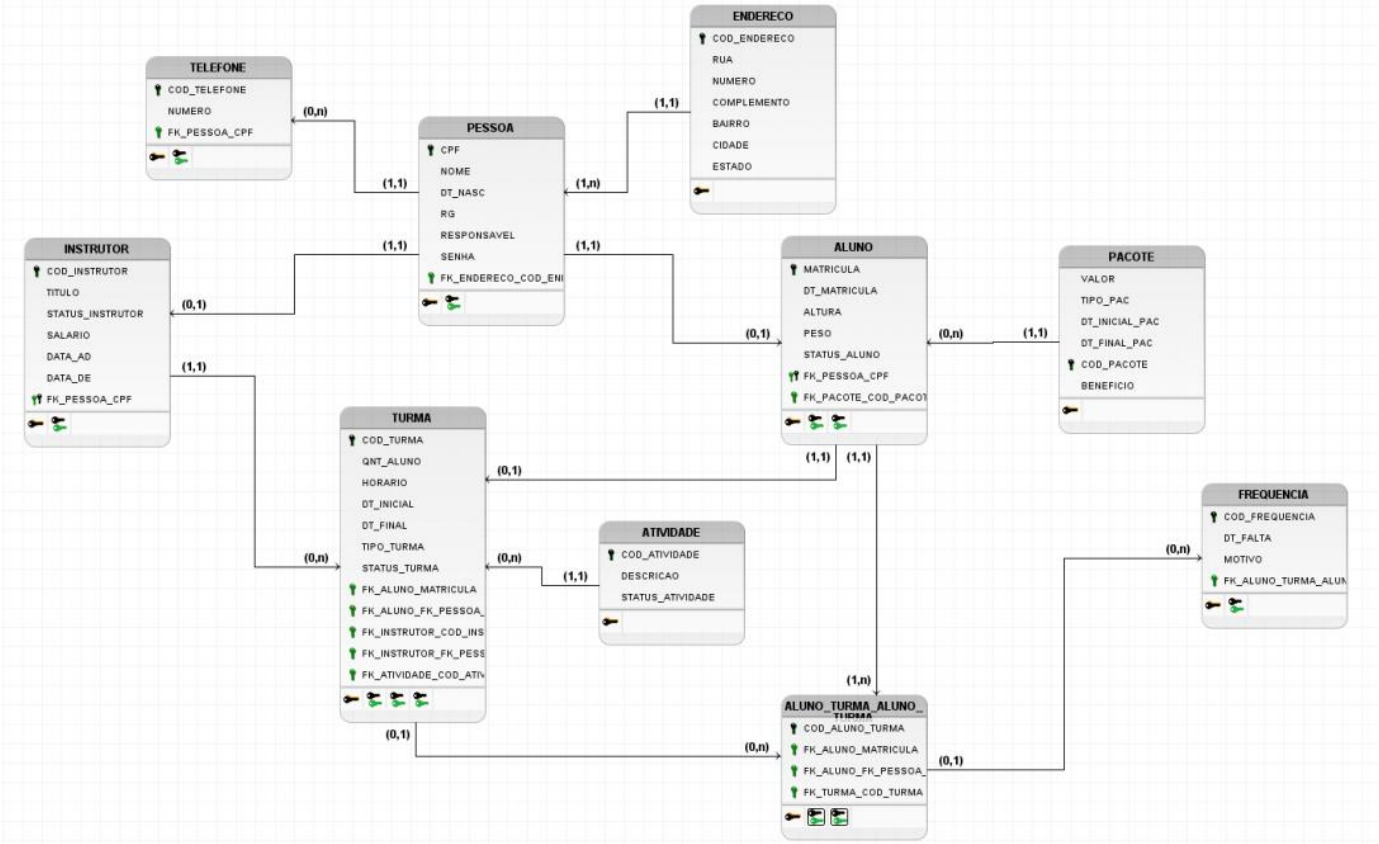
O Pacote Bronze é destinado às pessoas que desejam fazer somente uma atividade dentre as disponíveis na academia. O Pacote Bronze se aplica aos esportistas que desejam praticar até quatro atividades diferentes, escolhidas livremente. Já o Pacote Ouro dá livre acesso a todas as atividades disponíveis, nos horários que melhor se encaixem nas agendas dos praticantes. Ele é ideal para quem deseja praticar múltiplas atividades e ainda ter acesso a um grande desconto, em relação ao valor normalmente cobrado por cada atividade separadamente.

Além de tudo isso, outra inovação é a ferramenta que possibilita a criação de turmas com restrição de idade. Visando atender o público infantil, e permitindo assim uma atenção maior para as necessidades específicas dessa faixa etária. Facilitando o aprendizado dos mais novos, e permitindo aos instrutores a aplicação de atividades de forma uniforme e otimizada para esse público. Ao passo que também visa atender aos pais que precisam deixar os filhos em segurança enquanto praticam suas atividades.

MODELO CONCEITUAL



MODELO LÓGICO



DICIONÁRIO DE DADOS

Estrutura da tabela PESSOA

Coluna	Tipo	Aceita Nulos	PK / FK	Coluna FK (tabela origem)
CPF	char(14)	NÃO	PK	
NOME	varchar(50)	NÃO		
DT_NASC	DATE	NÃO		
RG	varchar(15)	NÃO		
RESPONSAVEL	bit	SIM		
NOME_RESPONSAVEL	varchar(50)	SIM		
COD_ENDERECO	smallint	NÃO	FK	COD_ENDERECO (Endereco)
SENHA	varchar(20)	NÃO		

Estrutura da tabela TELEFONE

Coluna	Tipo	Aceita Nulos	PK / FK	Coluna FK (tabela origem)
COD_TELEFONE	smallint	NÃO	PK	
NUMERO	char(15)	NÃO		
CPF_PESSOA	char(14)	NÃO	FK	CPF_PESSOA (Pessoa)

Estrutura da tabela ENDERECO

Coluna	Tipo	Aceita Nulos	PK / FK	Coluna FK (tabela origem)
COD_ENDERECO	smallint	NÃO	PK	
RUA	varchar(50)	NÃO		
NUMERO	varchar(5)	NÃO		
COMPLEMENTO	varchar(50)	SIM		
BAIRRO	varchar(30)	NÃO		
CIDADE	varchar(50)	NÃO		
ESTADO	char(2)	NÃO		

Estrutura da tabela INSTRUTOR

Coluna	Tipo	Aceita Nulos	PK / FK	Coluna FK (tabela origem)
COD_INSTRUTOR	smallint	NÃO	PK	
TITULO	varchar(50)	NÃO		
STATUS_INSTRUTOR	varchar(15)	NÃO		
CPF_PESSOA	char(14)	NÃO	FK	CPF_PESSOA (Pessoa)
DATA_AD	date	NÃO		
DATA_DE	date	SIM		
SALARIO	money	NÃO		

Estrutura da tabela ALUNO

Coluna	Tipo	Aceita Nulos	PK / FK	Coluna FK (tabela origem)
MATRICULA	smallint (1000,1)	NÃO	PK	
DT_MATRICULA	DATE	NÃO		
ALTURA	Decimal(3,2)	NÃO		
PESO	Decimal(5,2)	NÃO		
STATUS_ALUNO	BIT	NÃO		
CPF_PESSOA	char(14)	NÃO	FK	CPF_PESSOA (Pessoa)
COD_PACOTE	smallint	NÃO	FK	COD_PACOTE (Pacote)

Estrutura da tabela TURMA

Coluna	Tipo	Aceita Nulos	PK / FK	Coluna FK (tabela origem)
COD_TURMA	smallint	NÃO	PK	
QTD_ALUNO	smallint	NÃO		
HORARIO	TIME(0)	NÃO		
DT_INICIAL	DATE	NÃO		

DT_FINAL	DATE	NÃO		
TIPO_TURMA	varchar(8)	NÃO		
STATUS_TURMA	bit	NÃO		
COD_ATIVIDADE	smallint	NÃO	FK	COD_ATIVIDADE (Atividade)
COD_INSTRUTOR	smallint	NÃO	FK	COD_INSTRUTOR (Instrutor)
MATRICULA	smallint	SIM	FK	MATRICULA (Aluno)

Estrutura da tabela ALUNO_TURMA

Coluna	Tipo	Aceita Nulos	PK / FK	Coluna FK (tabela origem)
COD_ALUNO_TURMA	smallint	NÃO	PK	
MATRICULA	smallint	NÃO	FK	MATRICULA (Aluno)
COD_TURMA	smallint	NÃO	FK	COD_TURMA (Turma)

Estrutura da tabela FREQUENCIA

Coluna	Tipo	Aceita Nulos	PK / FK	Coluna FK (tabela origem)
COD_FREQUENCIA	smallint	NÃO	PK	
DT_FALTA	DATE	NÃO		
MOTIVO	varchar (100)	SIM		
COD_ALUNO_TURMA	smallint	NÃO	FK	COD_ALUNO_TURMA (Aluno_Turma)

Estrutura da tabela ATIVIDADE

Coluna	Tipo	Aceita Nulos	PK / FK	Coluna FK (tabela origem)
COD_ATIVIDADE	smallint	NÃO	PK	
DESCRICAO	varchar (20)	NÃO		
STATUS_ATIVIDADE	bit	NÃO		

Estrutura da tabela PACOTE

Coluna	Tipo	Aceita Nulos	PK / FK	Coluna FK (tabela origem)
COD_PACOTE	smallint	NÃO	PK	
VALOR	smallmoney	NÃO		
TIPO_PAC	varchar(20)	NÃO		
BENEFÍCIO	varchar(100)	NÃO		
DT_INICIAL_PAC	date	NÃO		
DT_FINAL_PAC	date	NÃO		

TERMO DE ABERTURA

Título Projeto:	**SGCAG (Sistema de Gerenciamento e Cadastramento de uma Academia de Ginástica)**
Cliente:	**UNIT**

1. Objetivo do Projeto

Com as pessoas se preocupando mais com sua saúde, estética e condicionamento físico, o mercado fitness vem crescendo bastante nos últimos anos. E com isso o negócio de academias sentiu a necessidade de um sistema de gerenciamento cujo objetivo é otimizar o processo de cadastro e gerenciamento de professores, alunos e turmas em sistemas de Academias de Ginástica. Dessa forma tornando rápida, por exemplo, a busca dos dados referentes aos planos de treinamento dos alunos. Substituindo assim as fichas de papel que atrasam as buscas, e ocupam desnecessariamente os que estão buscando tais informações.

2. Motivação/Necessidade do Negócio

O sistema foi desenvolvido baseado em pesquisas de mercado que mostram a dificuldade que muitas Academias de Ginástica têm em organizar o vasto número de dados que necessitam para funcionar da melhor forma. Um dos problemas comumente relatados pelos proprietários refere-se a dificuldade em se obter de forma ágil dados essenciais à realização das atividades diárias destas empresas. Como diferenciais o nosso sistema conta com três pacotes distintos de adesão. Sendo eles, o Pacote Bronze, Prata e o Ouro.

O Pacote Bronze é destinado às pessoas que desejam fazer somente uma atividade dentre as disponíveis na academia. O Pacote Prata se aplica aos esportistas que desejam praticar até quatro atividades diferentes, escolhidas livremente. Já o Pacote Ouro dá livre acesso a todas as atividades disponíveis, nos horários que melhor se encaixem nas agendas dos praticantes. Ele é ideal para quem deseja praticar múltiplas atividades e ainda ter acesso a um grande desconto, em relação ao valor normalmente cobrado por cada atividade separadamente.

Além de tudo isso, outra inovação é a ferramenta que possibilita a criação de turmas com restrição de idade. Visando atender o público infantil, e permitindo assim uma atenção maior para as necessidades específicas dessa faixa etária. Facilitando o aprendizado dos mais novos,

e permitindo aos instrutores a aplicação de atividades de forma uniforme e otimizada para esse público. Ao passo que também visa atender aos pais que precisam deixar os filhos em segurança enquanto praticam suas atividades.

3. Necessidades

Integração com sistemas e usuários desktop.

4. Riscos

O não conhecimento das tecnologias SQL Server e C#.

5. Critérios de Sucesso

Queremos alcançar com o software as Academias de Ginástica, visando tornar rápida a busca de dados referentes aos planos de treinamento dos alunos, substituindo as folhas de papel que atrasam a busca pelas informações. Além disso, possibilitando a criação de turmas para diversas idades, como crianças e idosos.

6. Restrições

Tempo de desenvolvimento, custos adicionais, não conhecimento em algumas tecnologias.

7. Gerente de Projetos

Auxiliar a equipe de T.I. sobre os requisitos, arquiteturas e plataformas, e delegar tarefas de acordo com o planejamento previamente elaborado.

8. Autorização

Recife, XX de Outubro de 2020

_____

XXXXXX XXXXXX

SCRIPT

```
CREATE DATABASE ACADEMIA
(EDITION = 'basic')
GO
```

```
CREATE DATABASE ACADEMIA
GO

USE ACADEMIA
GO

CREATE TABLE ENDERECO(
    COD_ENDERECO SMALLINT CONSTRAINT PK_ENDERECO PRIMARY KEY
IDENTITY,
    RUA VARCHAR(50) NOT NULL,
    NUMERO VARCHAR(5) NOT NULL,
    COMPLEMENTO VARCHAR(50),
    BAIRRO VARCHAR(30) NOT NULL,
    CIDADE VARCHAR(50) NOT NULL,
    ESTADO CHAR(2) NOT NULL
)
GO

CREATE TABLE PESSOA(
    CPF CHAR(14) CONSTRAINT PK_PESSOA PRIMARY KEY,
    NOME VARCHAR(50) NOT NULL,
    DT_NASC DATE NOT NULL,
    RG VARCHAR(15) NOT NULL,
    SENHA VARCHAR(20) NOT NULL,
    RESPONSAVEL BIT,
    COD_ENDERECO SMALLINT
    CONSTRAINT FK_PESSOA_ENDERECO
    FOREIGN KEY(COD_ENDERECO)
    REFERENCES ENDERECO(COD_ENDERECO)
)
GO

CREATE TABLE TELEFONE(
    COD_TELEFONE SMALLINT CONSTRAINT PK_TELEFONE_ACADEMIA PRIMARY
KEY IDENTITY,
    NUMERO CHAR(15) NOT NULL,
    CPF_PESSOA CHAR(14) NOT NULL,
    CONSTRAINT FK_TELEFONE_PESSOA
```

```
        FOREIGN KEY(CPF_PESSOA)
        REFERENCES PESSOA(CPF)
    )
    GO

    CREATE TABLE INSTRUTOR(
        COD_INSTRUTOR SMALLINT CONSTRAINT PK_INSTRUTOR_ACADEMIA PRIMARY
KEY IDENTITY,
        TITULO VARCHAR(50) NOT NULL,
        STATUS_INSTRUTOR VARCHAR(15) NOT NULL,
        DATA_AD DATE NOT NULL,
        DATA_DE DATE,
        SALARIO MONEY NOT NULL,
        CPF_PESSOA CHAR(14) NOT NULL,
        CONSTRAINT UQ_INSTRUTOR_PESSOA
        UNIQUE(CPF_PESSOA),
        CONSTRAINT FK_INSTRUTOR_PESSOA
        FOREIGN KEY(CPF_PESSOA)
        REFERENCES PESSOA(CPF)
    )
    GO

    CREATE TABLE PACOTE(
        COD_PACOTE SMALLINT CONSTRAINT PK_PACOTE_ACADEMIA PRIMARY KEY
IDENTITY,
        VALOR SMALLMONEY NOT NULL,
        TIPO_PAC VARCHAR(20) NOT NULL,
        DT_INICIAL_PAC DATE NOT NULL,
        DT_FINAL_PAC DATE NOT NULL,
        BENEFICIO CHAR(1) NOT NULL
    )
    GO

    CREATE TABLE ALUNO(
        MATRICULA SMALLINT CONSTRAINT PK_ALUNO_ACADEMIA PRIMARY KEY
IDENTITY(1000,1),
        DT_MATRICULA DATE NOT NULL,
        ALTURA DECIMAL(3,2) NOT NULL,
        PESO DECIMAL(5,2) NOT NULL,
        STATUS_ALUNO BIT NOT NULL,
        CPF_PESSOA CHAR(14) NOT NULL,
        CONSTRAINT UQ_ALUNO_PESSOA
```

```
        UNIQUE(CPF_PESSOA),
        CONSTRAINT FK_ALUNO_PESSOA
        FOREIGN KEY(CPF_PESSOA)
        REFERENCES PESSOA(CPF),
        COD_PACOTE SMALLINT NOT NULL,
        CONSTRAINT FK_ALUNO_PACOTE
        FOREIGN KEY(COD_PACOTE)
        REFERENCES PACOTE(COD_PACOTE)
    )
    GO

    CREATE TABLE ATIVIDADE(
        COD_ATIVIDADE SMALLINT CONSTRAINT PK_ATIVIDADE_ACADEMIA PRIMARY
KEY IDENTITY,
        DESCRICAO VARCHAR(20) NOT NULL,
        STATUS_ATIVIDADE BIT NOT NULL
    )
    GO

    CREATE TABLE TURMA(
        COD_TURMA SMALLINT CONSTRAINT PK_TURMA_ACADEMIA PRIMARY KEY
IDENTITY,
        QNT_ALUNO SMALLINT NOT NULL,
        HORARIO TIME(0) NOT NULL,
        DT_INICIAL DATE NOT NULL,
        DT_FINAL DATE NOT NULL,
        TIPO_TURMA VARCHAR(8) NOT NULL,
        CONSTRAINT CK_TURMA_TIPO
        CHECK(TIPO_TURMA IN('infantil', 'jovem', 'idoso')),
        STATUS_TURMA BIT NOT NULL, --
        COD_ATIVIDADE SMALLINT NOT NULL,
        CONSTRAINT FK_TURMA_ATIVIDADE
        FOREIGN KEY(COD_ATIVIDADE)
        REFERENCES ATIVIDADE(COD_ATIVIDADE),
        COD_INSTRUTOR SMALLINT NOT NULL,
        CONSTRAINT FK_TURMA_INSTRUTOR
        FOREIGN KEY (COD_INSTRUTOR)
        REFERENCES INSTRUTOR(COD_INSTRUTOR),
        MATRICULA SMALLINT,
        CONSTRAINT FK_TURMA_ALUNO
        FOREIGN KEY (MATRICULA)
        REFERENCES ALUNO(MATRICULA)
```

```
    )
    GO

    CREATE TABLE ALUNO_TURMA(
        COD_ALUNO_TURMA SMALLINT CONSTRAINT PK_ALUNO_TURMA_ACADEMIA
PRIMARY KEY IDENTITY,
        MATRICULA SMALLINT NOT NULL,
        CONSTRAINT FK_ALUNO_TURMA_ALUNO
        FOREIGN KEY (MATRICULA)
        REFERENCES ALUNO(MATRICULA),
        COD_TURMA SMALLINT NOT NULL,
        CONSTRAINT FK_ALUNO_TURMA_TURMA
        FOREIGN KEY (COD_TURMA)
        REFERENCES TURMA(COD_TURMA)
    )
    GO

    CREATE TABLE FREQUENCIA(
        COD_FREQUENCIA SMALLINT CONSTRAINT PK_FREQUENCIA_ACADEMIA
PRIMARY KEY IDENTITY,
        DT_FALTA DATE NOT NULL,
        MOTIVO VARCHAR(100),
        COD_ALUNO_TURMA SMALLINT NOT NULL,
        CONSTRAINT FK_FREQUENCIA_ALUNO_TURMA
        FOREIGN KEY (COD_ALUNO_TURMA)
        REFERENCES ALUNO_TURMA(COD_ALUNO_TURMA)
    )
    GO
```

CONSULTAS

1. STORED PROCEDURES

1.1 PROCEDURE + JOIN + FUNÇÃO DATA + DECLARAÇÃO DE VARIÁVEL + CONSULTA COM FILTRO + CONDICIONAL

```
--Como forma de estimular o aluno, seria o dono conseguir mostrar a evolução do mesmo de acordo com a sua estratégia, seja de emagrecimento,seja para ganho de massa muscular ou estabilidade

CREATE PROCEDURE uSP_EVOLUCAO_DO_PESO(
    @PESOATUAL DECIMAL(5,3),
    @CPF CHAR(14))
AS
BEGIN
    DECLARE
        @MES INT,
        @DATAMATRICULA DATE,
        @DATAMATRICULASTRING VARCHAR(10),
        @DIFERENCAPESO DECIMAL(5,3),
        @PESOINICIAL DECIMAL(5,3),
        @RESULTADO VARCHAR(20)

    SET @DATAMATRICULA = (SELECT A.DT_MATRICULA FROM ALUNO AS A INNER JOIN PESSOA AS P ON A.CPF_PESSOA = P.CPF WHERE CPF_PESSOA = @CPF)
    SET @DATAMATRICULASTRING  = (SELECT CONVERT(VARCHAR, @DATAMATRICULA,103))
-- Data convertida para o padrão dd/mm/aaaa
    SET @MES = DATEDIFF(MONTH, @DATAMATRICULA, GETDATE())
   SET @PESOINICIAL = (SELECT ALUNO.PESO FROM ALUNO INNER JOIN PESSOA ON ALUNO.CPF_PESSOA = PESSOA.CPF WHERE PESSOA.CPF = @CPF)
    SET @DIFERENCAPESO = @PESOINICIAL - @PesoAtual

    IF @PESOINICIAL > @PESOATUAL
    BEGIN
        SET @RESULTADO = CONCAT('EMAGRECEU ', ABS(@DIFERENCAPESO), ' KG')
    END

    IF @PESOINICIAL < @PESOATUAL
    BEGIN
        SET @RESULTADO = CONCAT('ENGORDOU ', ABS(@DIFERENCAPESO), ' KG')
    END

    IF @PESOINICIAL = @PESOATUAL
    BEGIN
```

```
        SET @RESULTADO = 'ESTÁVEL'
    END

    SELECT @DATAMATRICULASTRING AS 'DATA MATRÍCULA',
           @MES AS 'QTD MESES', @PESOINICIAL AS 'PESO INICIAL', @PESOATUAL AS
'PESO ATUAL',
           @RESULTADO AS 'RESULTADO' ;

END

EXEC uSP_EVOLUCAO_DO_PESO 82.80,'111.687.438-50'
GO
```

1.2 PROCEDURE + FUNÇÃO DATA + FUNÇÃO DE CONVERSÃO + FUNÇÃO DE TEXTO + CONSULTA COM FILTRO + DECLARAÇÃO DE VARIÁVEL

```
--Exibe um relatório com a quantidade de pacotes ativos no primeiro dia de cada
mês de um determinado ano.

CREATE PROCEDURE uSP_QUANTIDADE_DE_PACOTES_MENSAL(
@ANO VARCHAR(4)
)
AS
BEGIN
    DECLARE @JANEIRO INT , @FEVEREIRO INT,@MARCO INT,@ABRIL INT,@MAIO
INT,@JUNHO INT,@JULHO INT,@AGOSTO INT,
            @SETEMBRO INT, @OUTUBRO INT, @NOVEMBRO INT,@DEZEMBRO INT

    SET @JANEIRO = (SELECT COUNT(*) FROM PACOTE WHERE DT_INICIAL_PAC <
CONVERT(DATE, CONCAT(@ANO,'-01-01')) AND DT_FINAL_PAC > CONVERT(DATE,
CONCAT(@ANO,'-01-01')))
    SET @FEVEREIRO = (SELECT COUNT(*) FROM PACOTE WHERE DT_INICIAL_PAC <
CONVERT(DATE, CONCAT(@ANO,'-02-01')) AND DT_FINAL_PAC > CONVERT(DATE,
CONCAT(@ANO,'-02-01')))
    SET @MARCO = (SELECT COUNT(*) FROM PACOTE WHERE DT_INICIAL_PAC <
CONVERT(DATE, CONCAT(@ANO,'-03-01')) AND DT_FINAL_PAC > CONVERT(DATE,
CONCAT(@ANO,'-03-01')))
    SET @ABRIL = (SELECT COUNT(*) FROM PACOTE WHERE DT_INICIAL_PAC <
CONVERT(DATE, CONCAT(@ANO,'-04-01')) AND DT_FINAL_PAC > CONVERT(DATE,
CONCAT(@ANO,'-04-01')))
```

```
    SET @MAIO = (SELECT COUNT(*) FROM PACOTE WHERE DT_INICIAL_PAC < CONVERT(DATE, CONCAT(@ANO,'-05-01')) AND DT_FINAL_PAC > CONVERT(DATE, CONCAT(@ANO,'-05-01')))
    SET @JUNHO = (SELECT COUNT(*) FROM PACOTE WHERE DT_INICIAL_PAC < CONVERT(DATE, CONCAT(@ANO,'-06-01')) AND DT_FINAL_PAC > CONVERT(DATE, CONCAT(@ANO,'-06-01')))
    SET @JULHO = (SELECT COUNT(*) FROM PACOTE WHERE DT_INICIAL_PAC < CONVERT(DATE, CONCAT(@ANO,'-07-01')) AND DT_FINAL_PAC > CONVERT(DATE, CONCAT(@ANO,'-07-01')))
    SET @AGOSTO = (SELECT COUNT(*) FROM PACOTE WHERE DT_INICIAL_PAC < CONVERT(DATE, CONCAT(@ANO,'-08-01')) AND DT_FINAL_PAC > CONVERT(DATE, CONCAT(@ANO,'-08-01')))
    SET @SETEMBRO = (SELECT COUNT(*) FROM PACOTE WHERE DT_INICIAL_PAC < CONVERT(DATE, CONCAT(@ANO,'-09-01')) AND DT_FINAL_PAC > CONVERT(DATE, CONCAT(@ANO,'-09-01')))
    SET @OUTUBRO = (SELECT COUNT(*) FROM PACOTE WHERE DT_INICIAL_PAC < CONVERT(DATE, CONCAT(@ANO,'-10-01')) AND DT_FINAL_PAC > CONVERT(DATE, CONCAT(@ANO,'-10-01')))
    SET @NOVEMBRO = (SELECT COUNT(*) FROM PACOTE WHERE DT_INICIAL_PAC < CONVERT(DATE, CONCAT(@ANO,'-11-01')) AND DT_FINAL_PAC > CONVERT(DATE, CONCAT(@ANO,'-11-01')))
    SET @DEZEMBRO = (SELECT COUNT(*) FROM PACOTE WHERE DT_INICIAL_PAC < CONVERT(DATE, CONCAT(@ANO,'-12-01')) AND DT_FINAL_PAC > CONVERT(DATE, CONCAT(@ANO,'-12-01')))

    SELECT @JANEIRO AS QTD_JANEIRO, @FEVEREIRO AS QTD_FEVEREIRO, @MARCO AS QTD_MARCO, @ABRIL AS QTD_ABRIL, @MAIO AS QTD_MAIO, @JUNHO AS QTD_JUNHO,
    @JULHO AS QTD_JULHO, @AGOSTO AS QTD_AGOSTO, @SETEMBRO AS QTD_SETEMBRO, @OUTUBRO AS QTD_OUTUBRO, @NOVEMBRO AS QTD_NOVEMBRO, @DEZEMBRO AS QTD_DEZEMBRO
END

EXEC uSP_QUANTIDADE_DE_PACOTES_MENSAL '2020'
GO
```

1.3 PROCEDURE + DECLARAÇÃO DE VARIÁVEL + CONDICIONAL

```
-- EXIBE OS DADOS RELACIONADO AO SEU IMC, OBTENDO DADOS RELEVANTES SOBRE RISCOS E RECOMENDAÇÕES

CREATE PROC uSP_IMC
(
```

```
@PESO DECIMAL(5,2),
@ALTURA DECIMAL(3,2)
) AS
BEGIN
    DECLARE @IMC DECIMAL(5,2),
            @CLASSIFICACAO VARCHAR(30),
            @RISCO VARCHAR(100),
            @RECOMENDACAO VARCHAR(100)

    SET @IMC = @PESO / (@ALTURA * @ALTURA)

    IF @IMC < 18.5
    BEGIN
        SET @CLASSIFICACAO = 'Abaixo do peso'
        SET @RISCO = 'Queda capilar, alterações no sistema nervoso (depressão, irritabilidade), no sistema imunológico, debilidade física (fraqueza, falta de disposição) e problemas intestinais.'
        SET @RECOMENDACAO = 'Ingerir alimentos com muitos nutrientes e com pouca gordura saturada e gordura trans.'
    END
    IF @IMC > 18.6 AND @IMC < 24.9
    BEGIN
        SET @CLASSIFICACAO = 'Peso ideal'
        SET @RISCO = 'Fora de risco'
        SET @RECOMENDACAO = 'Manter peso dentro do normal, e deve fazer sempre os exames de rotina.'
    END
    IF @IMC > 25.0 AND @IMC < 29.9
    BEGIN
        SET @CLASSIFICACAO = 'Sobrepeso'
        SET @RISCO = 'Colesterol alto, diabetes do tipo 2, triglicérides altas, hipertensão etc.'
        SET @RECOMENDACAO = 'Dependendo da causa do sobrepeso, a pessoa precisa aumentar a rotina de exercícios juntamente com uma dieta balanceada, evitando alimentos como queijo prato, leite integral, manteiga etc.'
    END
    IF @IMC > 30.0 AND @IMC < 34.9
    BEGIN
        SET @CLASSIFICACAO = 'Obesidade grau 1'
        SET @RISCO = 'Risco moderado de: doenças cardiovasculares, doenças respiratórias, doenças gastrointestinais, diabetes tipo 2, alguns tipos de
```

```
cânceres, problemas ortopédicos, dores musculares,varizes nos membros inferiores, entre outros.'
        SET @RECOMENDACAO = 'Exercícios físicos, reeducação alimentar, e medicamentos com acompanhamento médico é fundamental para iniciar o processo de emagrecimento.'
    END
    IF @IMC > 35.0 AND @IMC < 39.9
    BEGIN
        SET @CLASSIFICACAO = 'Obesidade grau 2 (severa)'
        SET @RISCO = 'Risco moderado de: doenças cardiovasculares, doenças respiratórias, doenças gastrointestinais, diabetes tipo 2, alguns tipos de cânceres, problemas ortopédicos, dores musculares,varizes nos membros inferiores, entre outros.'
        SET @RECOMENDACAO = 'Exercícios físicos, reeducação alimentar, e medicamentos com acompanhamento médico é fundamental para iniciar o processo de emagrecimento.'
    END
    IF @IMC > 40.0
    BEGIN
        SET @CLASSIFICACAO = 'Obesidade grau 3 (mórbida)'
        SET @RISCO = 'Risco muito grave de: doenças cardiovasculares, doenças respiratórias, doenças gastrointestinais, diabetes tipo 2, alguns tipos de cânceres, problemas ortopédicos, dores musculares,varizes nos membros inferiores, entre outros.'
        SET @RECOMENDACAO = 'Exercícios físicos, reeducação alimentar, e medicamentos com acompanhamento médico é fundamental para iniciar o processo de emagrecimento.'
    END

    SELECT  @IMC AS "IMC",
            @CLASSIFICACAO AS "CLASSIFICAÇÃO",
            @RISCO AS "RISCO",
            @RECOMENDACAO AS "RECOMENDAÇÃO"
END
GO

exec uSP_IMC 70.0, 1.9
```

1.4 PROCEDURE + DECLARAÇÃO DE VARIÁVEL + CONSULTA COM FILTRO

```sql
-- CONSULTA A MOVIMENTAÇÃO FINANCEIRA DA ACADEMIA, SEUS GASTOS COM
FUNCIONÁRIOS E OS SEUS GANHOS COM SEUS ALUNOS

CREATE PROC uSP_SALDO
AS
BEGIN
    DECLARE @ENTRADA_BRONZE DECIMAL(7,2),
            @ENTRADA_PRATA DECIMAL(7,2),
            @ENTRADA_OURO DECIMAL(7,2),
            @ENTRADA_TOTAL DECIMAL(7,2),
            @SAIDA DECIMAL(7,2),
            @QTDE_ALUNOS INT,
            @QTDE_INTRUTORES INT

    SET @ENTRADA_BRONZE = (SELECT VALOR * QUANTIDADE FROM uV_PACOTE_INFO
WHERE PACOTE = 'Bronze')
    SET @ENTRADA_PRATA = (SELECT VALOR * QUANTIDADE FROM uV_PACOTE_INFO
WHERE PACOTE = 'Prata')
    SET @ENTRADA_OURO = (SELECT VALOR * QUANTIDADE FROM uV_PACOTE_INFO
WHERE PACOTE = 'Ouro')
    SET @ENTRADA_TOTAL = @ENTRADA_BRONZE + @ENTRADA_PRATA +
@ENTRADA_OURO
    SET @SAIDA = (SELECT SUM(SALARIO) FROM INSTRUTOR)
    SET @QTDE_ALUNOS = (SELECT COUNT(*) FROM ALUNO)
    SET @QTDE_INTRUTORES = (SELECT COUNT(*) FROM INSTRUTOR)

    SELECT  @QTDE_ALUNOS AS "Nº DE ALUNOS",
            @QTDE_INTRUTORES AS "Nº DE INSTRUTORES",
            @SAIDA AS "SAÍDA",
            @ENTRADA_TOTAL AS "ENTRADA",
            (@ENTRADA_TOTAL - @SAIDA) AS "SALDO"

END
GO

EXEC uSP_SALDO
```

1.5 PROCEDURE + JOIN + DECLARAÇÃO DE VARIÁVEL + CONSULTA COM FILTRO

```
-- Permite pegar 4 parâmetros sobre horários e datas disponíveis do aluno, cria uma pesquisa dentro das turmas existentes, nos retornando turmas que estão disponíveis ativas, ou turmas desativadas por falta de aluno dentro das limitações de horários e data imposta pelo aluno.

CREATE PROC uSP_TURMA_HORARIO
(
    @HORARIO_INICIAL TIME(0),
    @HORARIO_FINAL TIME(0),
    @INTERVALO_INICIAL DATE,
    @INTERVALO_FINAL DATE
)
AS
BEGIN

    SELECT
    T.HORARIO,
    T.TIPO_TURMA,
    T.DT_INICIAL,
    T.DT_FINAL,
    [dbo].[uF_STATUS](T.STATUS_TURMA) AS 'STATUS',
    A.DESCRICAO
    FROM
        TURMA AS T
    INNER JOIN ATIVIDADE AS A ON (T.COD_ATIVIDADE = A.COD_ATIVIDADE)
    WHERE (T.HORARIO BETWEEN @HORARIO_INICIAL AND @HORARIO_FINAL ) AND
(T.DT_INICIAL BETWEEN @INTERVALO_INICIAL AND @INTERVALO_FINAL)
END
GO

EXEC uSP_TURMA_HORARIO '08:00', '21:00', '2020/01/02', '2020/04/30'
```

1.6 PROCEDURE + JOIN + CONSULTA COM AGREGAÇÃO + CONSULTA COM FILTRO

```
-- PARA SABER O TOTAL DE FALTAS DOS ALUNOS EM UMA DETERMINADA TURMA E EM UM DETERMINADO INTERVALO DE TEMPO.
```

```sql
CREATE PROCEDURE uSP_FALTAS_ALUNOS_TURMA
(
@DATA_INICIO DATETIME2, -- Data na qual você deseja iniciar a pesquisa.
@DATA_FIM DATETIME2,    -- Até quando você quer que vá a pesquisa.
@TURMA SMALLINT          -- Código da turma na qual você deseja pesquisar as
faltas dos alunos.
)
AS
BEGIN
    SELECT
    T.COD_TURMA,
    ATIV.DESCRICAO,
    A.MATRICULA,
    P.NOME,
    COUNT(F.DT_FALTA) AS 'TOTAL DE FALTAS'
    FROM PESSOA AS P
        INNER JOIN ALUNO AS A
            ON P.CPF = A.CPF_PESSOA
        INNER JOIN ALUNO_TURMA AS AT
            ON A.MATRICULA = AT.MATRICULA
        INNER JOIN TURMA AS T
            ON AT.COD_TURMA = T.COD_TURMA
        INNER JOIN FREQUENCIA AS F
            ON F.COD_ALUNO_TURMA = AT.COD_ALUNO_TURMA
        INNER JOIN ATIVIDADE AS ATIV
            ON T.COD_ATIVIDADE = ATIV.COD_ATIVIDADE
    WHERE F.DT_FALTA BETWEEN @DATA_INICIO AND @DATA_FIM
    GROUP BY T.COD_TURMA, ATIV.DESCRICAO, A.MATRICULA, P.NOME
    HAVING T.COD_TURMA = @TURMA
END


EXEC uSP_FALTAS_ALUNOS_TURMA '2020-01-01','2020-06-30', 1
-- Retorna o total de faltas de todos os alunos no primeiro semestre da Turma
1.
```

1.7 PROCEDURE + CONSULTA COM FILTRO

```
-- PARA CONSULTAR TURMAS QUE INICIAM ENTRE AS DATAS CONSULTADAS

CREATE PROC uSP_DATA_TURMA(@DATA_INICIAL SMALLDATETIME, @DATA_FINAL
SMALLDATETIME)
AS
BEGIN
        SELECT T.COD_TURMA,
               T.HORARIO,
               T.TIPO_TURMA,
               T.DT_INICIAL
        FROM TURMA AS T
        WHERE DT_INICIAL BETWEEN @DATA_INICIAL AND @DATA_FINAL

END

EXEC uSP_DATA_TURMA '2020-04-02', '2020-09-05'
```

2. VIEWS

2.1 - VIEW + JOIN + FUNÇÃO DE AGREGAÇÃO

```
      CREATE VIEW uV_PACOTE_INFO AS
      SELECT  P.TIPO_PAC AS "PACOTE",
              P.[VALOR],
              DBO.uF_BENEFICIO(P.BENEFICIO) AS "BENEFÍCIOS",
              COUNT(*) AS "QUANTIDADE"
      FROM PACOTE AS P
          INNER JOIN ALUNO A
              ON P.[COD_PACOTE] = A.[COD_PACOTE]
      GROUP BY P.TIPO_PAC, P.[VALOR], P.BENEFICIO

      SELECT * FROM uV_PACOTE_INFO
```

2.2 - VIEW + JOIN

```
-- TURMA + INSTRUTOR + MONITOR + ENDEREÇO
```

```sql
CREATE VIEW uV_TURMA_INFO AS
SELECT  T.COD_TURMA ,
        T.QNT_ALUNO,
        T.HORARIO,
        T.TIPO_TURMA,
        DBO.uF_STATUS(T.STATUS_TURMA) AS "STATUS",
        IP.NOME AS "INSTRUTOR",
        (IE.RUA + ', nº ' + CAST(IE.NUMERO AS VARCHAR) + ' - ' +
        IE.BAIRRO + ' - ' + IE.CIDADE + ' - ' + IE.ESTADO) AS "ENDEREÇO
DO INSTRUTOR",
        AP.NOME AS "MONITOR",
        (AE.RUA + ', nº] ' + CAST(AE.NUMERO AS VARCHAR) + ' - ' +
        AE.BAIRRO + ' - ' + AE.CIDADE + ' - ' + AE.ESTADO) AS "ENDEREÇO
DO MONITOR"
FROM TURMA T
    INNER JOIN INSTRUTOR AS I
        ON T.COD_INSTRUTOR = I.COD_INSTRUTOR
    INNER JOIN PESSOA AS IP
        ON I.CPF_PESSOA = IP.CPF
    INNER JOIN ENDERECO AS IE
        ON IP.COD_ENDERECO = IE.COD_ENDERECO

    INNER JOIN ALUNO AS A
        ON T.MATRICULA = A.MATRICULA
    INNER JOIN PESSOA AS AP
        ON A.CPF_PESSOA = AP.CPF
    INNER JOIN ENDERECO AS AE
        ON AP.COD_ENDERECO = AE.COD_ENDERECO
GO

SELECT * FROM dbo.uV_TURMA_INFO
```

2.3 - VIEW + JOIN

```
-- ALUNO + VIEW(uV_TURMA_INFO)

CREATE VIEW uV_ALUNO_TURMA AS
SELECT  A.MATRICULA,
        P.NOME,
        A.[ALTURA],
        A.[PESO],
        T.HORARIO,
        T.TIPO_TURMA,
        ATV.DESCRICAO AS "ATIVIDADE",
        T.STATUS,
        T.INSTRUTOR AS "INSTRUTOR"
FROM ALUNO AS A
    INNER JOIN PESSOA AS P
        ON A.CPF_PESSOA = P.CPF
    INNER JOIN ALUNO_TURMA AS AT
        ON A.MATRICULA =  AT.MATRICULA
    INNER JOIN uV_TURMA_INFO AS T
        ON AT.COD_TURMA = T.COD_TURMA
    INNER JOIN ATIVIDADE AS ATV
        ON  ATV.COD_ATIVIDADE = T.COD_TURMA
GO

SELECT * FROM [dbo].[uV_ALUNO_TURMA]
```

2.4 - VIEW + JOIN

```
--View para ter todas as informações do aluno, podendo ter a
flexibilidade de ter as informações específicas mais rápido.

CREATE VIEW uV_ALUNO_INFO AS
SELECT
    P.CPF,
    P.NOME,
    P.DT_NASC AS 'NASCIMENTO',
    P.RG,
    PA.TIPO_PAC AS 'TIPO',
    A.PESO,
    A.ALTURA,
    ED.ESTADO,
    ED.CIDADE,
    ED.BAIRRO,
    ED.RUA,
    ED.NUMERO,
    T.NUMERO AS 'TELEFONE'
FROM
    ALUNO AS A
INNER JOIN PESSOA AS P ON (A.CPF_PESSOA = P.CPF)
INNER JOIN ENDERECO AS ED ON (P.COD_ENDERECO = ED.COD_ENDERECO)
INNER JOIN TELEFONE AS T ON (P.CPF = T.CPF_PESSOA)
INNER JOIN PACOTE AS PA ON (A.COD_PACOTE = PA.COD_PACOTE)
```

2.4 - VIEW + JOIN + FUNÇÃO DE AGREGAÇÃO

```
-- -View - Retorna o número de alunos por atividade, cujo objetivo é ter
um relatório das atividades mais frequentadas, e estimular de alguma forma
a adesão por outras turmas

CREATE VIEW uV_ALUNO_ATIVIDADE AS
SELECT COUNT(*) AS "QUANTIDADE DE ALUNOS",
       ATIV.DESCRICAO AS "ATIVIDADE"
FROM PESSOA AS P
```

```
    INNER JOIN ALUNO AS A
        ON P.CPF = A.CPF_PESSOA
    INNER JOIN ALUNO_TURMA AS AT
        ON A.MATRICULA = AT.MATRICULA
    INNER JOIN TURMA AS T
        ON AT.COD_TURMA = T.COD_TURMA
    INNER JOIN ATIVIDADE AS ATIV
        ON  ATIV.COD_ATIVIDADE = T.COD_ATIVIDADE
GROUP BY ATIV.DESCRICAO
SELECT * FROM uV_ALUNO_ATIVIDADE
GO
```

2.5 - VIEW + JOIN

```
-- RETORNA UMA TABELA COM INFORMAÇÕES BÁSICAS DO ALUNO (NOME E MATRÍCULA)
E CORRESPONDENTES À SUA TURMA E PACOTE CONTRATADO.

CREATE VIEW uV_ALUNO_TURMA_PAC AS
SELECT TOP 1000 P.NOME,
        A.MATRICULA,
        T.TIPO_TURMA,
        PAC.TIPO_PAC
FROM PESSOA AS P
    INNER JOIN ALUNO AS A
        ON P.CPF = A.CPF_PESSOA
    INNER JOIN ALUNO_TURMA AS AT
        ON A.MATRICULA = AT.MATRICULA
    INNER JOIN TURMA AS T
        ON AT.COD_TURMA = T.COD_TURMA
    INNER JOIN PACOTE AS PAC
        ON PAC.COD_PACOTE = A.COD_PACOTE
ORDER BY P.NOME


SELECT * FROM uV_ALUNO_TURMA_PAC
GO
```

2.6 - VIEW + JOIN

```
-- MOSTRAR MATRÍCULA, NOME, DATA DA FALTA, MOTIVO DA FALTA E CÓDIGO DA TURMA DO ALUNO

CREATE VIEW uV_ALUNO_FREQUENCIA AS
SELECT A.MATRICULA,
       P.NOME,
       F.DT_FALTA,
       F.MOTIVO,
       AT.COD_TURMA
FROM ALUNO AS A
    INNER JOIN PESSOA AS P
        ON A.CPF_PESSOA = P.CPF
    INNER JOIN ALUNO_TURMA AS AT
        ON A.MATRICULA = AT.MATRICULA
    INNER JOIN FREQUENCIA AS F
        ON F.COD_FREQUENCIA = AT.COD_ALUNO_TURMA
    GO

SELECT * FROM [dbo].[uV_ALUNO_FREQUENCIA]
```

3. TRIGGERS

3.1 - TRIGGER + FUNÇÃO DE DATA

```
--VERIFICA SE O USUÁRIO QUE ESTÁ SENDO CADASTRADO É DE MENOR, E NÃO SINALIZOU NO CADASTRO QUE TEM RESPONSÁVEL. TAMBÉM, VERIFICA SE HÁ ALGUM INSTRUTOR DE MENOR, POIS NÃO É PERMITIDO EXERCER A FUNÇÃO DE INSTRUTOR SENDO MENOR DE IDADE.

CREATE TRIGGER uTR_MENOR_IDADE
ON DBO.PESSOA
FOR INSERT AS
    DECLARE @DT_NASC DATE,
            @RESP BIT
```

```
    SELECT @DT_NASC = I.DT_NASC, @RESP = I.RESPONSAVEL FROM INSERTED I

    IF ( (DATEDIFF(DAY, @DT_NASC, GETDATE()) / 365) < 18 AND @RESP = 0)
    BEGIN
        RAISERROR('O aluno é de menor, favor adicionar os dados
referentes ao responsável!', 16, 1)
        ROLLBACK TRANSACTION
    END

    IF ((DATEDIFF(DAY, @DT_NASC, GETDATE()) / 365) < 18 AND @RESP IS
NULL)
    BEGIN
        RAISERROR('O instrutor não pode ser de menor',16, 1)
        ROLLBACK TRANSACTION
    END
GO
```

3.2 - TRIGGER + DECLARAÇÃO DE VARIÁVEL + CONDICIONAL + FILTRO

```
-- É REALIZADO UM CONTROLE SOBRE A QUANTIDADE DE ALUNOS NA TURMA, PARA
QUE SEJA INCREMENTADO OU DECREMENTADO ASSIM QUE SURGIR UM NOVO ALUNO,
TROCA E SAÍDA DE ALUNO DA TURMA

CREATE TRIGGER uTR_CONTROLE_TURMA
ON DBO.ALUNO_TURMA
AFTER INSERT, UPDATE, DELETE AS
BEGIN
    DECLARE @COD_ANT SMALLINT,
            @COD_NOV SMALLINT

    IF EXISTS (SELECT * FROM DELETED) AND EXISTS (SELECT * FROM
INSERTED)
    BEGIN
        SET @COD_ANT = (SELECT COD_TURMA FROM DELETED)
```

```
        SET @COD_NOV = (SELECT COD_TURMA FROM INSERTED)

        UPDATE TURMA
        SET QNT_ALUNO = QNT_ALUNO - 1
        WHERE COD_TURMA = @COD_ANT

        UPDATE TURMA
        SET QNT_ALUNO = QNT_ALUNO + 1
        WHERE COD_TURMA = @COD_NOV
    END
    IF EXISTS (SELECT * FROM DELETED) AND NOT EXISTS (SELECT * FROM
INSERTED)
    BEGIN
        SET @COD_ANT = (SELECT COD_TURMA FROM DELETED)

        UPDATE TURMA
        SET QNT_ALUNO = QNT_ALUNO - 1
        WHERE COD_TURMA = @COD_ANT
    END
    ELSE
    BEGIN
        SET @COD_NOV = (SELECT COD_TURMA FROM INSERTED)

        UPDATE TURMA
        SET QNT_ALUNO = QNT_ALUNO + 1
        WHERE COD_TURMA = @COD_NOV
    END
END
GO
```

3.3 - TRIGGER + DECLARAÇÃO DE VARIÁVEL + CONDICIONAL + FILTRO

```
--Trigger para controlar a matrícula do aluno em novas turmas quando ele
já chegou na quantidade máxima de turmas do seu pacote.
```

```
CREATE TRIGGER tgr_QUANTIDADE_ALUNO_TURMAS ON dbo.ALUNO_TURMA
FOR INSERT AS
BEGIN
    DECLARE @QTDE INT,
            @MATRICULA INT,
            @TIPO VARCHAR(20)
    SET @QTDE = (SELECT
        COUNT(P.COD_PACOTE) AS 'QUANTIDADE'
    FROM
        ALUNO_TURMA AS AT
        INNER JOIN ALUNO AS A ON (A.MATRICULA = AT.MATRICULA)
        INNER JOIN PACOTE AS P ON (P.COD_PACOTE = A.COD_PACOTE)
        GROUP BY A.MATRICULA
        HAVING A.MATRICULA = (SELECT MATRICULA FROM inserted))
    SET @MATRICULA = (SELECT MATRICULA FROM inserted)
    SET @TIPO = (SELECT * FROM dbo.uFN_TIPO_MATRICULA(@MATRICULA))

    IF (@TIPO = 'Bronze')
    BEGIN
        IF(@QTDE > 1)
        BEGIN
        RAISERROR('O aluno está matriculado na quantidade máxima que seu
pacote permite.', 16, 1)
        ROLLBACK TRANSACTION
        END
    END
    IF (@TIPO = 'Prata')
    BEGIN
        IF(@QTDE > 4)
        BEGIN
        RAISERROR('O aluno está matriculado na quantidade máxima que seu
pacote permite.', 16, 1)
        ROLLBACK TRANSACTION
        END
    END
END
```

```
GO
```

4. FUNCTIONS

4.1 - FUNCTION + CONDICIONAL

```
-- REALIZA A CONVERSÃO DO STATUS QUE NO BANCO ESTÁ COMO BIT, TRANSFORMANDO O MESMO EM TEXTO COMO, POR EXEMPLO, 'DESATIVADO' OU 'ATIVA'.
CREATE FUNCTION uF_STATUS
(
@STATUS BIT
)
RETURNS VARCHAR(10) AS
BEGIN
    DECLARE @RESULT VARCHAR(10)

    IF @STATUS = 0
    BEGIN
        SET @RESULT = 'Desativado'
    END
    ELSE
    BEGIN
        SET @RESULT = 'Ativa'
    END

    RETURN @RESULT
END
GO
```

4.2 - FUNCTION + FUNÇÃO DE TEXTO + FUNÇÃO DE DATA + DECLARAÇÃO DE VARIÁVEL + CONDICIONAL

```
-- PODE REALIZAR A VERIFICAÇÃO DE HÁ QUANTO TEMPO O INSTRUTOR TRABALHA NA ACADEMIA OU, NO CASO DE DEMISSÃO, QUANTO TEMPO ELE TRABALHOU ATÉ A SUA DEMISSÃO.

CREATE FUNCTION uFN_TEMPO_CORRIDO
```

```
(
@DataInicial DATE,
@DataFinal DATE
)
RETURNS VARCHAR(40)
AS
BEGIN
    DECLARE
        @ANO INT,
        @MES INT,
        @DIAS INT

    SET @ANO = DATEDIFF(DAY, @DataInicial, @DataFinal) / 365
    SET @MES = DATEDIFF(MONTH, (DATEADD(YEAR, @ANO, @DataInicial)),
@DataFinal)

    IF @MES = 12
    BEGIN
        SET @MES = 11
    END

    SET @DIAS = DATEDIFF(DAY, DATEADD(MONTH, @MES,DATEADD(YEAR, @ANO,
@DataInicial)), @DataFinal)

    IF @ANO = 0
    BEGIN
        RETURN CONCAT(@MES, ' meses', ' e ', ABS(@DIAS), ' dias')
    END
    IF @DIAS = 0 AND @MES = 0
    BEGIN
        RETURN CONCAT(@ANO, ' anos')
    END
    IF @MES = 0
    BEGIN
        RETURN CONCAT(@ANO, ' anos', ' e ', ABS(@DIAS), ' dias')
    END
```

```
    IF @DIAS = 0
    BEGIN
        RETURN CONCAT(@ANO, ' anos', ' e ', @MES, ' meses')
    END

    RETURN CONCAT(@ANO, ' anos ', @MES, ' meses', ' e ', ABS(@DIAS), '
dias')
END
GO

SELECT  P.NOME AS "INSTRUTOR", DBO.uFN_TEMPO_CORRIDO(I.DATA_AD,
GETDATE()) AS "TEMPO TRABALHADO" FROM INSTRUTOR AS I
      INNER JOIN PESSOA P
            ON I.CPF_PESSOA = P.CPF
```

4.3 - FUNCTION + CONDICIONAL

```
-- RECEBE O CÓDIGO REFERENTE AO BENEFÍCIO, E RETORNA O TEXTO REFERENTE
AOS MESMO.

CREATE FUNCTION uF_BENEFICIO
(
@BENEFICIO CHAR(1)
)
RETURNS VARCHAR(100) AS
BEGIN
    IF(@BENEFICIO = 1)
    BEGIN
        RETURN 'O Pacote Bronze é destinado às pessoas que desejam fazer
somente uma atividade dentre as disponíveis na academia.'
    END
    IF(@BENEFICIO = 2)
    BEGIN
        RETURN 'O Pacote Prata se aplica aos esportistas que desejam
praticar até quatro atividades diferentes, escolhidas livremente.'
    END
```

```
    IF(@BENEFICIO = 3)
    BEGIN
        RETURN 'O Pacote Ouro de livre acesso a todas as atividades disponíveis, nos horários que melhor se encaixem nas agendas dos praticantes, acesso a um grande desconto, em relação ao valor normalmente cobrado por cada atividade separadamente. '
    END

    RETURN 'Código de benefício inválido!'
END
GO
```

4.4 - FUNCTION + JOIN

```
-- RETORNA UMA TABELA COM A LOCALIZAÇÃO E O CONTATO DO ALUNO/RESPONSÁVEL CASO PRECISAR CONTATAR O MESMO

CREATE FUNCTION uFN_ALUNO_CONTATO(@NOME VARCHAR(100))
RETURNS TABLE
AS
RETURN(
    SELECT P.NOME, EN.RUA, EN.BAIRRO, EN.CIDADE, EN.ESTADO, TEL.NUMERO AS TELEFONE
    FROM [dbo].[PESSOA] P
    INNER JOIN ENDERECO EN
    ON EN.[COD_ENDERECO] = P.[COD_ENDERECO]
    INNER JOIN TELEFONE TEL
    ON P.[CPF]= TEL.[CPF_PESSOA]
    WHERE P.NOME LIKE (@NOME + '%')

)
GO
```

4.5 - FUNCTION TABLE + JOIN + CONSULTA COM AGREGAÇÃO + CONSULTA COM FILTRO

```
--Retorna o número de instrutores por atividade, de forma que seja possível remanejá-los, caso possuam skill de outras atividades, a fim de dar suporte a turmas que estão desfalcadas. Além disso, o dono consegue ter noção da faixa salarial dos mesmos.

CREATE FUNCTION uFN_INSTRUTOR_POR_ATIVIDADE_FAIXASALARIAL
(@SALARIO money)
RETURNS TABLE
AS
RETURN
(
SELECT COUNT(*) AS "QUANTIDADE DE INSTRUTORES",
                ATIV.DESCRICAO AS "ATIVIDADE",
                I.SALARIO AS SALARIO
FROM PESSOA AS P
    INNER JOIN INSTRUTOR AS I
        ON P.CPF = I.CPF_PESSOA
    INNER JOIN TURMA AS T
        ON I.COD_INSTRUTOR = T.COD_INSTRUTOR
    INNER JOIN ATIVIDADE AS ATIV
        ON  ATIV.COD_ATIVIDADE = T.COD_ATIVIDADE
GROUP BY ATIV.DESCRICAO, I.SALARIO
HAVING I.SALARIO > @SALARIO
)
GO
SELECT * FROM uFN_INSTRUTOR_POR_ATIVIDADE_FAIXASALARIAL(10000)
```

4.6 - FUNCTION TABLE + JOIN + CONSULTA COM AGREGAÇÃO + CONSULTA COM FILTRO

```
-- Function table que retorna o dado do tipo de pacote de uma matrícula passada por parâmetro.
```

```
CREATE FUNCTION uFN_TIPO_MATRICULA
(
@MATRICULA INT
)
RETURNS TABLE
AS
RETURN
(
SELECT
    P.TIPO_PAC
FROM ALUNO AS A
INNER JOIN PACOTE AS P ON (P.COD_PACOTE = A.COD_PACOTE)
WHERE A.MATRICULA = @MATRICULA
)
GO
```

5. OUTRAS CONSULTAS

5.1 - CONSULTA COM FILTRO

```
-- CONSULTA PARA SABER QUAIS ALUNOS ATIVOS TÊM FALTAS NÃO JUSTIFICADAS

SELECT * FROM PESSOA, ALUNO, ATIVIDADE, FREQUENCIA
    WHERE FREQUENCIA.MOTIVO IS NULL AND STATUS_ATIVIDADE = '1' AND
STATUS_ALUNO = '1'
ORDER BY NOME
```