

PROVA P1

Grupo:

Guilherme Marinho Bernardi – 0020482323025

1. Disserte sobre os tipos de dados utilizados no SQL. Exemplifique

R. No SQL Server, utilizam-se diversos tipos de dados, de acordo com a necessidade da coluna a ser armazenada. Podemos dividir os dados nas seguintes categorias:

TIPOS NUMÉRICOS	
Nome	Descrição
INT	Número inteiro (32 bits)
BIGINT	Número inteiro (64 bits)
SMALLINT	Número inteiro (16 bits)
TINYINT	Número inteiro (8 bits)
DECIMAL	Número decimal com precisão fixa
FLOAT	Número decimal de ponto flutuante

Os tipos numéricos são usados quando deseja-se armazenar um número na coluna.

NOTA -> Os tipos que estão grifados são os mais utilizados. Os demais, são “variantes” que são utilizadas muitas vezes para otimizar bancos de dados.

TIPOS DE TEXTO	
Nome	Descrição
VARCHAR (N M)	Cadeia de caracteres de tamanho N a M (variável)
CHAR (N)	Cadeia de caracteres de tamanho N (fixo)
TEXT	Cadeia de caracteres sem tamanho definido (Usado para textos longos)

Os tipos de texto são usados quando deseja-se armazenar variados textos na coluna.

TIPOS DE TEMPO	
Nome	Descrição
DATE	Armazena data (sem hora)
TIME	Armazena hora (Sem data)
DATETIME	Armazena data e hora

Os tipos de tempo são usados quando deseja-se armazenar hora ou data na coluna.

Estes são os principais dados em SQL Server, a maioria das colunas são preenchidas com estes. Alguns outros tipos interessantes são:

OUTROS TIPOS	
Nome	Descrição
BIT	Armazena um valor booleano
XML	Armazena dados XML
TEMP	Indica que um dado é temporário (armazenado apenas na sessão)

Vale ressaltar que há mais tipos de dados, menos utilizados, em SQL. Estes podem ser melhor analisados na referência [1], documentação do SQL Server.

2. Qual a diferença entre os comandos T-SQL FORMAT e CONVERT?

Tanto o FORMAT quanto o CONVERT são utilizados para a edição de colunas do tipo de 'datas'. Ambas irão modificar a formatação do retorno do tipo date, mas não alteram o banco de dados.

O comando CONVERT realiza a conversão da data para um tipo passado como argumento, e para uma formatação passada numericamente (como mostra a tabela abaixo):

CONVERT	
Número do argumento	Resultado
1	mm/dd/yy
101	mm/dd/yyyy
2	yy.mm.dd
102	yyyy.mm.dd
3	dd/mm/yy
103	dd/mm/yyyy

OBS: Para mais padrões numéricos, conferir documentação [2]

Exemplo de implementação do convert:

```

1 SELECT CONVERT(datetime, DtAdmissao, 2) AS DataConvertida
2 from Empregado
3
4 select DtAdmissao as DataOriginal
5 from Empregado

```

100 %

Resultados Mensagens

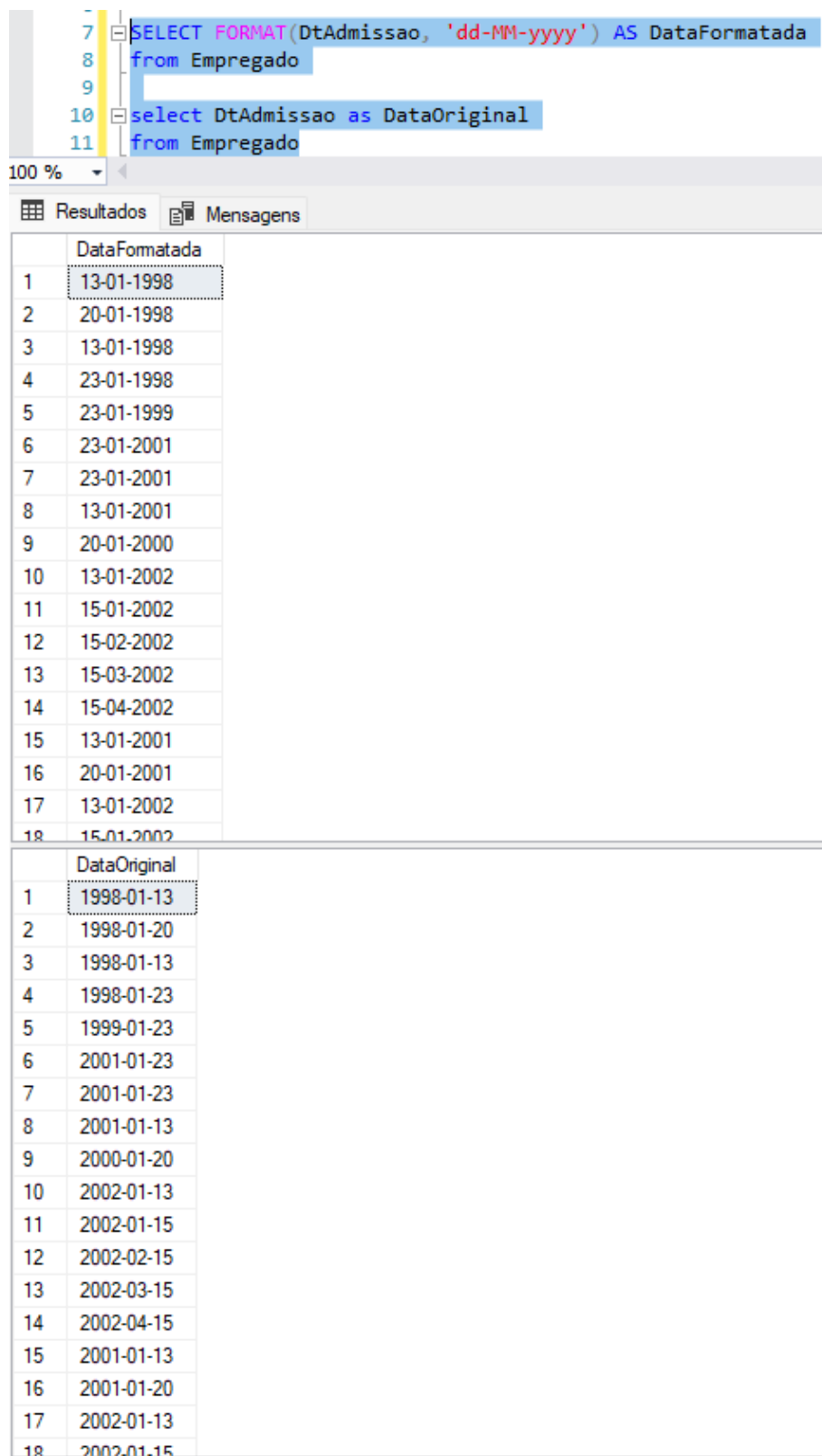
	DataConvertida
1	1998-01-13 00:00:00.000
2	1998-01-20 00:00:00.000
3	1998-01-13 00:00:00.000
4	1998-01-23 00:00:00.000
5	1999-01-23 00:00:00.000
6	2001-01-23 00:00:00.000
7	2001-01-23 00:00:00.000
8	2001-01-13 00:00:00.000
9	2000-01-20 00:00:00.000
10	2002-01-13 00:00:00.000
11	2002-01-15 00:00:00.000
12	2002-02-15 00:00:00.000
13	2002-03-15 00:00:00.000
14	2002-04-15 00:00:00.000
15	2001-01-13 00:00:00.000
16	2001-01-20 00:00:00.000
17	2002-01-13 00:00:00.000
18	2002-01-15 00:00:00.000

	DataOriginal
1	1998-01-13
2	1998-01-20
3	1998-01-13
4	1998-01-23
5	1999-01-23
6	2001-01-23
7	2001-01-23
8	2001-01-13
9	2000-01-20
10	2002-01-13
11	2002-01-15
12	2002-02-15
13	2002-03-15
14	2002-04-15
15	2001-01-13
16	2001-01-20
17	2002-01-13
18	2002-01-15

NOTE: Como originalmente o tipo era 'date', e foi solicitado a conversão para 'datetime', a parte horária ficou com seu padrão default.

O FORMAT é utilizado quando o analista deseja passar como argumento a maneira com que quer que a data seja retornada, o que torna o comando mais flexível, porém menos eficiente.

Exemplo de implementação:



```

7 SELECT FORMAT(DtAdmissao, 'dd-MM-yyyy') AS DataFormatada
8 from Empregado
9
10 select DtAdmissao as DataOriginal
11 from Empregado

```

	DataFormatada
1	13-01-1998
2	20-01-1998
3	13-01-1998
4	23-01-1998
5	23-01-1999
6	23-01-2001
7	23-01-2001
8	13-01-2001
9	20-01-2000
10	13-01-2002
11	15-01-2002
12	15-02-2002
13	15-03-2002
14	15-04-2002
15	13-01-2001
16	20-01-2001
17	13-01-2002
18	15-01-2002

	DataOriginal
1	1998-01-13
2	1998-01-20
3	1998-01-13
4	1998-01-23
5	1999-01-23
6	2001-01-23
7	2001-01-23
8	2001-01-13
9	2000-01-20
10	2002-01-13
11	2002-01-15
12	2002-02-15
13	2002-03-15
14	2002-04-15
15	2001-01-13
16	2001-01-20
17	2002-01-13
18	2002-01-15

3. Qual a diferença entre JOIN, INNER JOIN e UNION?

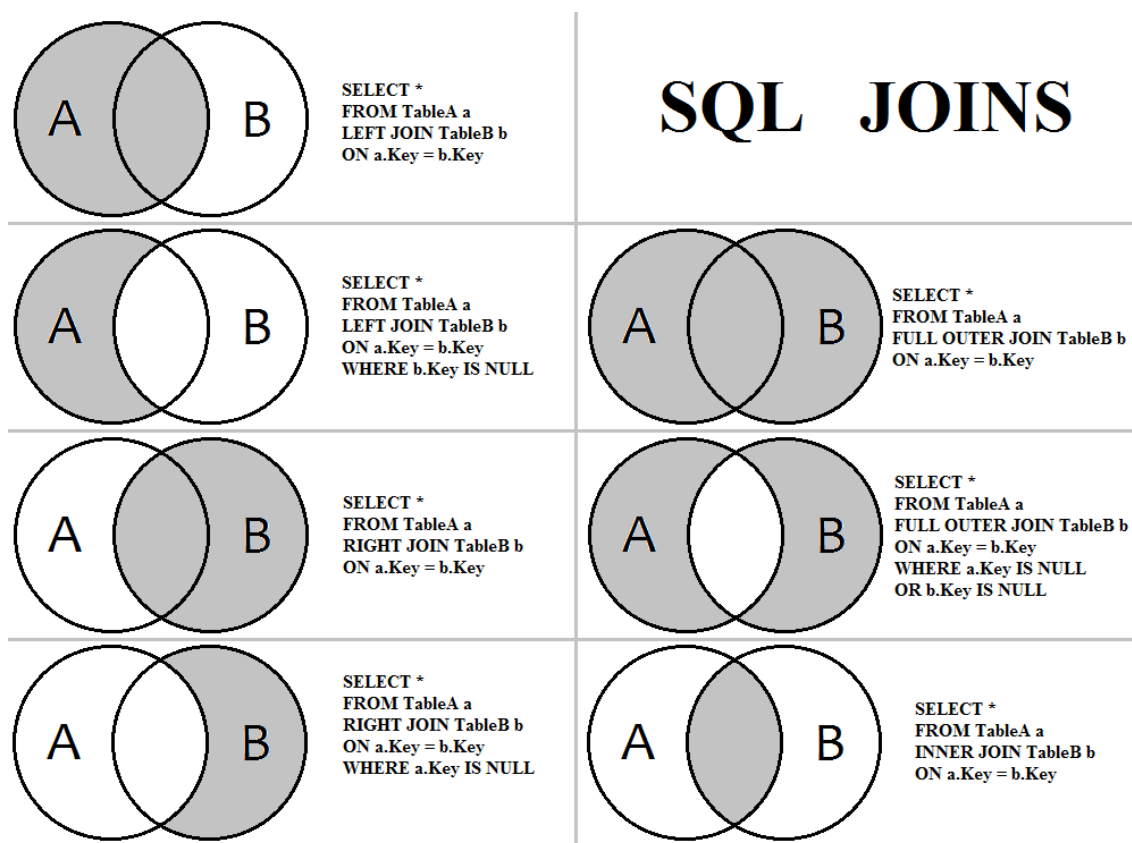
‘Join’, ‘Inner Join’, e ‘Union’ são operações SQL para a combinação de diferentes tabelas e dados, que possuem relação entre si

‘Join’ referencia-se apenas a uma ligação entre as tabelas, por isso, costuma ser usado com algum argumento : LEFT/RIGHT/OUTER/FULL OUTER.

‘Inner Join’ liga apenas as tabelas num determinado ponto onde haja compatibilidade de dados entre ambas (uma chave estrangeira = uma chave primária). É exatamente o que o SQL retorna caso o ‘Join’ não seja usado com Left/Right/etc.

‘Union’ retorna todas as combinações entre duas tabelas, já eliminando duplicadas, numa única lista.

A imagem abaixo exemplifica graficamente cada utilização, (fonte [3]):



Exemplo no SQL Server:

Tabelas originais:

1	SELECT * from Empregado
2	SELECT * from Depto

100 %

Resultados Mensagens

	NrEmpregado	NmEmpregado	NrGerente	DsCargo	NrDepto	DtAdmissao	VSalario	VComissao
1	100	FHC	NULL	Presidente	1	1998-01-13	20000	NULL
2	101	Ivete Sangalo	NULL	Secretária	20	1998-01-20	1500	NULL
3	110	Chico Buarque	NULL	Gerente	10	1998-01-13	2500	0
4	111	Romário	110	Operário	10	1998-01-23	900	800
5	112	Rivaldo	110	Operário	10	1999-01-23	900	1700
6	113	Dida	110	Operário	10	2001-01-23	900	600
7	114	Ronaldinho	110	Operário	10	2001-01-23	900	1800
8	120	Caetano Velozo	NULL	Gerente	20	2001-01-13	3000	0
9	121	Marisa Monte	120	Secretária	20	2000-01-20	700	NULL
10	130	Milton Nascimento	NULL	Gerente	30	2002-01-13	2000	500
11	131	Ana Paula Arósio	130	Vendedora	30	2002-01-15	500	2200
12	132	Adriana Esteves	130	Vendedora	30	2002-02-15	500	2500
13	133	Regina Duarte	130	Vendedora	30	2002-03-15	500	1500
14	134	Glória Pires	130	Vendedora	30	2002-04-15	500	1000
15	140	Gilberto Gil	NULL	Gerente	40	2001-01-13	3000	0
16	141	Marina Lima	140	Secretária	40	2001-01-20	700	NULL
17	150	Zé Ramalho	NULL	Gerente	50	2002-01-13	2000	0
18	151	Camila Pitanga	150	Vendedora	50	2002-01-15	600	1500
19	152	Deborah Secco	150	Vendedora	50	2002-01-15	600	1000

	NrDepto	NmDepto	DsLocal	VOrcamento
1	1	Presidência	São Paulo	20000
2	10	Fábrica	Santo André	30000
3	20	RH	São Paulo	30000
4	30	Comercial	São Paulo	60000
5	40	Infomática	Santo André	40000
6	50	Telemarketing	São Paulo	20000
7	60	Administrativo	Curitiba	8000

A) Utilizando o JOIN (LEFT)

Retornando apenas departamentos com empregados alocados:

```
SELECT *
from Empregado left join Depto
on Empregado.NrDepto = Depto.NrDepto
```

	NrEmpregado	NmEmpregado	NrGerente	DsCargo	NrDepto	DtAdmissao	VSalario	VComissao	NrDepto	NmDepto	DsLocal	VOrcamento
1	100	FHC	NULL	Presidente	1	1998-01-13	20000	NULL	1	Presidência	São Paulo	20000
2	101	Ivete Sangalo	NULL	Secretária	20	1998-01-20	1500	NULL	20	RH	São Paulo	30000
3	110	Chico Buarque	NULL	Gerente	10	1998-01-13	2500	0	10	Fábrica	Santo André	30000
4	111	Romário	110	Operário	10	1998-01-23	900	800	10	Fábrica	Santo André	30000
5	112	Rivaldo	110	Operário	10	1999-01-23	900	1700	10	Fábrica	Santo André	30000
6	113	Dida	110	Operário	10	2001-01-23	900	600	10	Fábrica	Santo André	30000
7	114	Ronaldinho	110	Operário	10	2001-01-23	900	1800	10	Fábrica	Santo André	30000
8	120	Caetano Velozo	NULL	Gerente	20	2001-01-13	3000	0	20	RH	São Paulo	30000
9	121	Marisa Monte	120	Secretária	20	2000-01-20	700	NULL	20	RH	São Paulo	30000
10	130	Milton Nascimento	NULL	Gerente	30	2002-01-13	2000	500	30	Comercial	São Paulo	60000
11	131	Ana Paula Arósio	130	Vendedora	30	2002-01-15	500	2200	30	Comercial	São Paulo	60000
12	132	Adriana Esteves	130	Vendedora	30	2002-02-15	500	2500	30	Comercial	São Paulo	60000
13	133	Regina Duarte	130	Vendedora	30	2002-03-15	500	1500	30	Comercial	São Paulo	60000
14	134	Glória Pires	130	Vendedora	30	2002-04-15	500	1000	30	Comercial	São Paulo	60000
15	140	Gilberto Gil	NULL	Gerente	40	2001-01-13	3000	0	40	Infomática	Santo André	40000
16	141	Marina Lima	140	Secretária	40	2001-01-20	700	NULL	40	Infomática	Santo André	40000
17	150	Zé Ramalho	NULL	Gerente	50	2002-01-13	2000	0	50	Telemarketing	São Paulo	20000
18	151	Camila Pitanga	150	Vendedora	50	2002-01-15	600	1500	50	Telemarketing	São Paulo	20000
19	152	Deborah Secco	150	Vendedora	50	2002-01-15	600	1000	50	Telemarketing	São Paulo	20000

B) Utilizando o INNER JOIN

Retornando apenas departamentos com funcionários operários, e seus nomes

```
SELECT NmDepto, NmEmpregado, DsCargo
FROM Depto inner join Empregado
on Depto.NrDepto = Empregado.NrDepto
where DsCargo = 'Operário'
```

	NmDepto	NmEmpregado	DsCargo
1	Fábrica	Romário	Operário
2	Fábrica	Rivaldo	Operário
3	Fábrica	Dida	Operário
4	Fábrica	Ronaldinho	Operário

C) Utilizando Union:

(No banco de dados acima não faria sentido utilizar este comando pois usamos quando duas tabelas possuem colunas que devem ser unidas, ou seja, possuem a mesma estrutura, mas ficaria da seguinte maneira:)

1	SELECT	NmEmpregado, NrGerente	FROM	Empregado
2	UNION			
3	SELECT	DsLocal, VlOrcamento	FROM	Depto

100 %

Resultados Mensagens

	NmEmpregado	NrGerente
1	Adriana Esteves	130
2	Ana Paula Arósio	130
3	Caetano Velozo	NULL
4	Camila Pitanga	150
5	Chico Buarque	NULL
6	Curitiba	8000
7	Deborah Secco	150
8	Dida	110
9	FHC	NULL
10	Gilberto Gil	NULL
11	Glória Pires	130
12	Ivete Sangalo	NULL
13	Marina Lima	140
14	Marisa Monte	120
15	Milton Nascimento	NULL
16	Regina Duarte	130
17	Rivaldo	110
18	Romário	110
19	Ronaldinho	110
20	Santo André	30000
21	Santo André	40000
22	São Paulo	20000
23	São Paulo	30000
24	São Paulo	60000
25	Zé Ramalho	NULL

4. Conforme as tabelas abaixo:

FUNCIONARIOS	
PK	IdFuncionario
	NomeFuncionario
	DtAdmissao
	IdCargo
	VISalario
	IdSecao
	DtUltFeria
	DtDemissao
	CPF

SECAO	
PK	IdDepto
PK	IdDivisao
PK	IdSecao
	NomeSecao

CARGO	
PK	IdCargo
	Descr_cargo
	IdcargoConfia

DEPARTAMENTO	
PK	IdDepto
	NomeDepto

DIVISAO	
PK	IdDepto
PK	IdDivisao
	NomeDivisao

PROJETO	
PK	CodProj
	SiglaProj
	DtInicio
	DtFim
	StatusProj

FUNC PROJ	
PK	CodProj
PK	IdFuncionario
	DtAlocIni
	DtAlocFim

Defina o código para sua criação garantindo a integridade referencial do seu modelo. Defina o código para inserção de registros nas tabelas, observando que a tabela FUNCIONARIOS contenha no mínimo 5 (cinco) funcionários de cada seção

Código :

```
1 CREATE TABLE DEPARTAMENTO
2 (
3     IdDepto INT PRIMARY KEY,
4     NomeDepto VARCHAR(255) NOT NULL,
5 );
6
7 CREATE TABLE PROJETO
8 (
9     CodProj INT PRIMARY KEY,
10    SiglaProj VARCHAR(25) NOT NULL,
11    DtInicio DATE NOT NULL,
12    DtFim DATE,
13    StatusProj VARCHAR(25) NOT NULL,
14 );
15
16 CREATE TABLE DIVISAO
17 (
18     IdDepto INT NOT NULL,
19     IdDivisao INT NOT NULL,
20     NomeDivisao VARCHAR(255),
21     PRIMARY KEY (IdDepto, IdDivisao),
22     CONSTRAINT Fk_IdDepto FOREIGN KEY (IdDepto) REFERENCES DEPARTAMENTO (IdDepto),
23 );
24
```

```

42 CREATE TABLE FUNCIONARIOS
43 (
44     IdFuncionario INT PRIMARY KEY,
45     NomeFuncionario VARCHAR(255) NOT NULL,
46     DtAdmissao DATE NOT NULL,
47     IdCargo INT NOT NULL,
48     VlSalario FLOAT NOT NULL,
49     IdSecao INT NOT NULL,
50     IdDivisao INT NOT NULL,
51     IdDepto INT NOT NULL, --NOTA -> Como Seção possui chave primária composta tripla, só pode
52     DtUltFeria DATE NOT NULL, --haver uma FK com 3 valores, sendo necessário armazenar IdDepto e IdDivisao
53     DtDemissao DATE,
54     CPF INT NOT NULL,
55     CONSTRAINT Fk_IdDepto_Fk_IdDiv_Fk_IdSecao FOREIGN KEY (IdDepto, IdDivisao, IdSecao) REFERENCES SECAO (IdDepto, IdDivisao, IdSecao),
56     CONSTRAINT Fk_IdCargo FOREIGN KEY (IdCargo) REFERENCES CARGO (IdCargo),
57 );
58
25 CREATE TABLE SECAO
26 (
27     IdDepto INT NOT NULL,
28     IdDivisao INT NOT NULL,
29     IdSecao INT NOT NULL,
30     NomeSecao VARCHAR(255) NOT NULL,
31     PRIMARY KEY (IdDepto, IdDivisao, IdSecao),
32     CONSTRAINT Fk_IdDepto_Fk_IdDivisao FOREIGN KEY (IdDepto, IdDivisao) REFERENCES DIVISAO (IdDepto, IdDivisao),
33 );
34
35 CREATE TABLE CARGO
36 (
37     IdCargo INT PRIMARY KEY,
38     Descr_Cargo TEXT NOT NULL,
39     IdCargoConfia INT NOT NULL,
40 );
41
CREATE TABLE FUNC_PROJ
(
    CodProj INT NOT NULL,
    IdFuncionario INT NOT NULL,
    DtAlocIni DATE NOT NULL,
    DtAlocFim DATE,
    PRIMARY KEY (CodProj, IdFuncionario),
    CONSTRAINT Fk_IdFuncionario FOREIGN KEY (IdFuncionario) REFERENCES FUNCIONARIOS (IdFuncionario),
    CONSTRAINT Fk_CodProj FOREIGN KEY (CodProj) REFERENCES PROJETO (CodProj),
);

```

Abaixo, segue o Código para ser copiado e testado:

COMANDOS CREATE TABLE

```

CREATE TABLE DEPARTAMENTO
(
    IdDepto INT PRIMARY KEY,
    NomeDepto VARCHAR(255) NOT NULL,
);

CREATE TABLE PROJETO
(
    CodProj INT PRIMARY KEY,
    SiglaProj VARCHAR(25) NOT NULL,
    DtInicio DATE NOT NULL,
    DtFim DATE,
    StatusProj VARCHAR(25) NOT NULL,
);

CREATE TABLE DIVISAO
(
    IdDepto INT NOT NULL,
    IdDivisao INT NOT NULL,
    NomeDivisao VARCHAR(255),
    PRIMARY KEY (IdDepto, IdDivisao),
    CONSTRAINT Fk_IdDepto FOREIGN KEY (IdDepto) REFERENCES DEPARTAMENTO (IdDepto),
);

CREATE TABLE SECAO
(
    IdDepto INT NOT NULL,
    IdDivisao INT NOT NULL,
    IdSecao INT NOT NULL,
    NomeSecao VARCHAR(255) NOT NULL,
    PRIMARY KEY (IdDepto, IdDivisao, IdSecao),
    CONSTRAINT Fk_IdDepto_Fk_IdDivisao FOREIGN KEY (IdDepto, IdDivisao) REFERENCES DIVISAO (IdDepto, IdDivisao),
);

CREATE TABLE CARGO
(
    IdCargo INT PRIMARY KEY,
    Descr_Cargo TEXT NOT NULL,
    IdCargoConfia INT NOT NULL,
);

CREATE TABLE FUNCIONARIOS
(
    IdFuncionario INT PRIMARY KEY,
    NomeFuncionario VARCHAR(255) NOT NULL,
    DtAdmissao DATE NOT NULL,
    IdCargo INT NOT NULL,
    VlSalario FLOAT NOT NULL,
    IdSecao INT NOT NULL,
    IdDivisao INT NOT NULL,
    IdDepto INT NOT NULL, --NOTA -> Como Seção possui chave primária composta tripla, só pode
    DtUltFeria DATE NOT NULL, --haver uma FK com 3 valores, sendo necessário armazenar IdDepto e
    IdDivisao
    DtDemissao DATE,
    CPF INT NOT NULL,
    CONSTRAINT Fk_IdDepto_FK_IdDiv_FK_IdSecao FOREIGN KEY (IdDepto, IdDivisao, IdSecao) REFERENCES
    SECAO (IdDepto, IdDivisao, IdSecao),
    CONSTRAINT Fk_IdCargo FOREIGN KEY (IdCargo) REFERENCES CARGO (IdCargo),
);

```

```
CREATE TABLE FUNC_PROJ
(
    CodProj INT NOT NULL,
    IdFuncionario INT NOT NULL,
    DtAlocIni DATE NOT NULL,
    DtAlocFim DATE,
    PRIMARY KEY (CodProj, IdFuncionario),
    CONSTRAINT Fk_IdFuncionario FOREIGN KEY (IdFuncionario) REFERENCES FUNCIONARIOS (IdFuncionario),
    CONSTRAINT Fk_CodProj FOREIGN KEY (CodProj) REFERENCES PROJETO (CodProj),
);
```

Agora, adicionando os valores, temos:

```

71
72 INSERT INTO DEPARTAMENTO
73     VALUES (10, 'Administracao')
74 INSERT INTO DEPARTAMENTO
75     VALUES (20, 'Tecnologia')
76 INSERT INTO DEPARTAMENTO
77     VALUES (30, 'RH')
78 INSERT INTO DEPARTAMENTO
79     VALUES (40, 'Gerencia')
80 SELECT * FROM DEPARTAMENTO

```

.20 %

	IdDepto	NomeDepto
1	10	Administracao
2	20	Tecnologia
3	30	RH
4	40	Gerencia

```

82 | INSERT INTO PROJETO
83 |     VALUES(10, 'ContratacaoAnual', '17/08/2023', NULL, 'ATIVO')
84 | INSERT INTO PROJETO
85 |     VALUES(20, 'RelatorioMensal', '10/10/2023', '18/10/2023', 'COMPLETO')
86 | INSERT INTO PROJETO
87 |     VALUES(30, 'SistemaPIX', '17/09/2023', NULL, 'ATIVO')
88 | SELECT * FROM PROJETO

```

%				
Resultados Mensagens				
CodProj	SiglaProj	DtInicio	DtFim	StatusProj
10	ContratacaoAnual	2023-08-17	NULL	ATIVO
20	RelatorioMensal	2023-10-10	2023-10-18	COMPLETO
30	SistemaPIX	2023-09-17	NULL	ATIVO

```

94  INSERT INTO DIVISAO
95      VALUES (10, 001, 'Alfa')
96  INSERT INTO DIVISAO
97      VALUES (10, 002, 'Beta')
98  INSERT INTO DIVISAO
99      VALUES (10, 003, 'Gama')
100 INSERT INTO DIVISAO
101     VALUES (20, 011, 'Bravo')
102 INSERT INTO DIVISAO
103     VALUES (20, 012, 'Kilo')
104 INSERT INTO DIVISAO
105     VALUES (20, 013, 'Romeo')
106 INSERT INTO DIVISAO
107     VALUES (30, 101, 'Nit')
108 INSERT INTO DIVISAO
109     VALUES (40, 200, 'Operacional')
110 INSERT INTO DIVISAO
111     VALUES (40, 201, 'Gerencial')
112 SELECT * FROM DIVISAO
  
```

120 %

	IdDepto	IdDivisao	NomeDivisao
1	10	1	Alfa
2	10	2	Beta
3	10	3	Gama
4	20	11	Bravo
5	20	12	Kilo
6	20	13	Romeo
7	30	101	Nit
8	40	200	Operacional
9	40	201	Gerencial


```
114 |
115 | INSERT INTO SECAO
116 |     VALUES (10, 001, 900, 'Secao Inicial')
117 | INSERT INTO SECAO
118 |     VALUES (10, 001, 901, 'Secao Carlos')
119 | INSERT INTO SECAO
120 |     VALUES (10, 001, 902, 'Secao Ana')
121 | INSERT INTO SECAO
122 |     VALUES (10, 001, 903, 'Secao Fábio')
123 | INSERT INTO SECAO
124 |     VALUES (10, 002, 900, 'Secao Jones')
125 | INSERT INTO SECAO
126 |     VALUES (10, 002, 901, 'Secao Sara')
127 | INSERT INTO SECAO
128 |     VALUES (10, 003, 900, 'Secao Manuela')
129 | INSERT INTO SECAO
130 |     VALUES (20, 11, 800, 'Secao Marianita')
131 | INSERT INTO SECAO
132 |     VALUES (20, 11, 801, 'Secao Roberto')
133 | INSERT INTO SECAO
134 |     VALUES (20, 12, 800, 'Secao Paulo')
135 | INSERT INTO SECAO
136 |     VALUES (20, 13, 800, 'Secao Guilherme')
137 | INSERT INTO SECAO
138 |     VALUES (30, 101, 700, 'Secao Gabriel')
139 | INSERT INTO SECAO
140 |     VALUES (30, 101, 701, 'Secao Andrea')
```

```
142 |     VALUES (40, 200, 600, 'Secao Romero')
143 | INSERT INTO SECAO
144 |     VALUES (40, 200, 601, 'Secao Giovinnazzi')
145 | INSERT INTO SECAO
146 |     VALUES (40, 200, 602, 'Secao Ricciardo')
147 | INSERT INTO SECAO
148 |     VALUES (40, 201, 600, 'Secao Vettel')
149 | INSERT INTO SECAO
150 |     VALUES (40, 201, 601, 'Secao Alonso')
151 | SELECT* FROM SECAO
```

151 | SELECT* FROM SECAO

120 %

Resultados Mensagens

	IdDepto	IdDivisao	IdSecao	NomeSecao
1	10	1	900	Secao Inicial
2	10	1	901	Secao Carlos
3	10	1	902	Secao Ana
4	10	1	903	Secao Fábio
5	10	2	900	Secao Jones
6	10	2	901	Secao Sara
7	10	3	900	Secao Manuela
8	20	11	800	Secao Marianita
9	20	11	801	Secao Roberto
10	20	12	800	Secao Paulo
11	20	13	800	Secao Guilherme
12	30	101	700	Secao Gabriel
13	30	101	701	Secao Andrea
14	40	200	600	Secao Romero
15	40	200	601	Secao Giovinnazzi
16	40	200	602	Secao Ricciardo
17	40	201	600	Secao Vettel
18	40	201	601	Secao Alonso

```

152
153 INSERT INTO CARGO
154     VALUES(555, 'Gerente de Ativos', 555)
155 INSERT INTO CARGO
156     VALUES(444, 'Consultor financeiro', 444)
157 INSERT INTO CARGO
158     VALUES(333, 'Programador', 333)
159 INSERT INTO CARGO
160     VALUES(222, 'Analista de Redes', 222)
161 INSERT INTO CARGO
162     VALUES(111, 'Chefe', 111)
163
164 SELECT * FROM CARGO
165

```

120 %

Resultados Mensagens

	IdCargo	Descr_Cargo	IdCargoConfia
1	111	Chefe	111
2	222	Analista de Redes	222
3	333	Programador	333
4	444	Consultor financeiro	444
5	555	Gerente de Ativos	555

```

16/
168 INSERT INTO FUNCIONARIOS
169     VALUES (1, 'Ayrton', '03/09/2022', 111, 5020.30, 900,1,10, '18/10/2023', NULL, 6547824)
170 INSERT INTO FUNCIONARIOS
171     VALUES (2, 'Fabio', '03/09/2022', 222, 3020.30, 903,1,10, '09/01/2023', NULL, 254858)
172 INSERT INTO FUNCIONARIOS
173     VALUES (3, 'Ana', '03/11/2021', 333, 2020.30, 800,11,20, '09/10/2022', '10/10/2023', 5421878)
174 INSERT INTO FUNCIONARIOS
175     VALUES (4, 'Roberto', '08/10/2021', 444, 6054.30, 700,101,30, '02/04/2023', NULL, 3167441)
176 INSERT INTO FUNCIONARIOS
177     VALUES (5, 'Moreno', '05/09/2022', 555, 3333.30, 600,201,40, '16/04/2023', NULL, 316040)
178 INSERT INTO FUNCIONARIOS
179     VALUES (6, 'Tania', '15/09/2020', 333, 1000.30, 601,201,40, '18/08/2023', '10/10/2023', 324663)
180 SELECT * FROM FUNCIONARIOS

```

120 %

Resultados Mensagens

	IdFuncionario	NomeFuncionario	DtAdmissao	IdCargo	VLSalario	IdSecao	IdDivisao	IdDepto	DtUltFeria	DtDemissao	CPF
1	1	Ayrton	2022-09-03	111	5020.3	900	1	10	2023-10-18	NULL	6547824
2	2	Fabio	2022-09-03	222	3020.3	903	1	10	2023-01-09	NULL	254858
3	3	Ana	2021-11-03	333	2020.3	800	11	20	2022-10-09	2023-10-10	5421878
4	4	Roberto	2021-10-08	444	6054.3	700	101	30	2023-04-02	NULL	3167441
5	5	Moreno	2022-09-05	555	3333.3	600	201	40	2023-04-16	NULL	316040
6	6	Tania	2020-09-15	333	1000.3	601	201	40	2023-08-18	2023-10-10	324663

```

91 INSERT INTO FUNC_PROJ
92     VALUES(10, 1, '10/10/2023', null)
93 INSERT INTO FUNC_PROJ
94     VALUES(10, 2, '10/10/2023', null)
95 INSERT INTO FUNC_PROJ
96     VALUES(20, 3, '10/10/2023', null)
97 INSERT INTO FUNC_PROJ
98     VALUES(20, 4, '10/10/2023', null)
99 INSERT INTO FUNC_PROJ
100    VALUES(30, 5, '10/10/2023', null)
101 INSERT INTO FUNC_PROJ
102    VALUES(30, 6, '10/10/2023', null)

```

108 %

	CodProj	IdFuncionario	DtAlocIni	DtAlocFim
1	10	1	2023-10-10	NULL
2	10	2	2023-10-10	NULL
3	20	3	2023-10-10	NULL
4	20	4	2023-10-10	NULL
5	30	5	2023-10-10	NULL
6	30	6	2023-10-10	NULL

O Código se encontra abaixo:

CÓDIGO INSERTS

```

INSERT INTO DEPARTAMENTO
VALUES (10, 'Administracao')
INSERT INTO DEPARTAMENTO
VALUES (20, 'Tecnologia')
INSERT INTO DEPARTAMENTO
VALUES (30, 'RH')
INSERT INTO DEPARTAMENTO
VALUES (40, 'Gerencia')
SELECT * FROM DEPARTAMENTO

```

```

INSERT INTO PROJETO
VALUES(10, 'ContratacaoAnual', '17/08/2023', NULL, 'ATIVO')
INSERT INTO PROJETO
VALUES(20, 'RelatorioMensal', '10/10/2023', '18/10/2023', 'COMPLETO')
INSERT INTO PROJETO
VALUES(30, 'SistemaPIX', '17/09/2023', NULL, 'ATIVO')
SELECT * FROM PROJETO

```

```

INSERT INTO DIVISAO
VALUES (10, 001, 'Alfa')

```

```
INSERT INTO DIVISAO
VALUES (10, 002, 'Beta')
INSERT INTO DIVISAO
VALUES (10, 003, 'Gama')
INSERT INTO DIVISAO
VALUES (20, 011, 'Bravo')
INSERT INTO DIVISAO
VALUES (20, 012, 'Kilo')
INSERT INTO DIVISAO
VALUES (20, 013, 'Romeo')
INSERT INTO DIVISAO
VALUES (30, 101, 'Nit')
INSERT INTO DIVISAO
VALUES (40, 200, 'Operacional')
INSERT INTO DIVISAO
VALUES (40, 201, 'Gerencial')
SELECT * FROM DIVISAO

INSERT INTO SECAO
VALUES (10, 001, 900, 'Secao Inicial')
INSERT INTO SECAO
VALUES (10, 001, 901, 'Secao Carlos')
INSERT INTO SECAO
VALUES (10, 001, 902, 'Secao Ana')
INSERT INTO SECAO
VALUES (10, 001, 903, 'Secao Fábio')
INSERT INTO SECAO
VALUES (10, 002, 900, 'Secao Jones')
INSERT INTO SECAO
VALUES (10, 002, 901, 'Secao Sara')
INSERT INTO SECAO
VALUES (10, 003, 900, 'Secao Manuela')
INSERT INTO SECAO
VALUES (20, 11, 800, 'Secao Marianita')
INSERT INTO SECAO
VALUES (20, 11, 801, 'Secao Roberto')
INSERT INTO SECAO
VALUES (20, 12, 800, 'Secao Paulo')
INSERT INTO SECAO
VALUES (20, 13, 800, 'Secao Guilherme')
INSERT INTO SECAO
VALUES (30, 101, 700, 'Secao Gabriel')
INSERT INTO SECAO
VALUES (30, 101, 701, 'Secao Andrea')
INSERT INTO SECAO
VALUES (40, 200, 600, 'Secao Romero')
INSERT INTO SECAO
VALUES (40, 200, 601, 'Secao Giovinazzi')
INSERT INTO SECAO
VALUES (40, 200, 602, 'Secao Ricciardo')
INSERT INTO SECAO
VALUES (40, 201, 600, 'Secao Vettel')
INSERT INTO SECAO
VALUES (40, 201, 601, 'Secao Alonso')
SELECT* FROM SECAO

INSERT INTO CARGO
VALUES(555, 'Gerente de Ativos', 555)
INSERT INTO CARGO
VALUES(444, 'Consultor financeiro', 444)
```

```
INSERT INTO CARGO
VALUES(333, 'Programador', 333)
INSERT INTO CARGO
VALUES(222, 'Analista de Redes', 222)
INSERT INTO CARGO
VALUES(111, 'Chefe', 111)
SELECT * FROM CARGO

INSERT INTO FUNCIONARIOS
VALUES (1, 'Ayrton', '03/09/2022', 111, 5020.30, 900,1,10, '18/10/2023', NULL, 6547824)
INSERT INTO FUNCIONARIOS
VALUES (2, 'Fabio', '03/09/2022', 222, 3020.30, 903,1,10, '09/01/2023', NULL, 254858)
INSERT INTO FUNCIONARIOS
VALUES (3, 'Ana', '03/11/2021', 333, 2020.30, 800,11,20, '09/10/2022', '10/10/2023', 5421878)
INSERT INTO FUNCIONARIOS
VALUES (4, 'Roberto', '08/10/2021', 444, 6054.30, 700,101,30, '02/04/2023', NULL, 3167441)
INSERT INTO FUNCIONARIOS
VALUES (5, 'Moreno', '05/09/2022', 555, 3333.30, 600,201,40, '16/04/2023', NULL, 316040)
INSERT INTO FUNCIONARIOS
VALUES (6, 'Tania', '15/09/2020', 333, 1000.30, 601,201,40, '18/08/2023', '10/10/2023', 324663)
SELECT * FROM FUNCIONARIOS

INSERT INTO FUNC_PROJ
VALUES(10, 1, '10/10/2023', null)
INSERT INTO FUNC_PROJ
VALUES(10, 2, '10/10/2023', null)
INSERT INTO FUNC_PROJ
VALUES(20, 3, '10/10/2023', null)
INSERT INTO FUNC_PROJ
VALUES(20, 4, '10/10/2023', null)
INSERT INTO FUNC_PROJ
VALUES(30, 5, '10/10/2023', null)
INSERT INTO FUNC_PROJ
VALUES(30, 6, '10/10/2023', null)
SELECT * FROM FUNC_PROJ
```

5. Crie no modelo do exercício 4 uma nova tabela que identifique os ramais de telefone que o funcionário atende (respeitando regras de integridade) e Defina função para que recebendo parte do nome do funcionário retorne nome do funcionário, Nome da Seção e no máximo dois ramais do funcionário.

Primeiramente, criando a tabela de ATENDIMENTOS e inserindo dados:

```

196 CREATE TABLE ATENDIMENTOS
197 (
198     RamalPrincipal INT PRIMARY KEY,
199     RamalSecundario INT,
200     IdFuncionario INT NOT NULL,
201     CONSTRAINT Fk_IdFunc FOREIGN KEY (IdFuncionario) REFERENCES FUNCIONARIOS (IdFuncionario),
202 );
203

```

```

204 INSERT INTO ATENDIMENTOS
205     VALUES (123456789, null, 1)
206 INSERT INTO ATENDIMENTOS
207     VALUES (987654321, 1123456, 2)
208 INSERT INTO ATENDIMENTOS
209     VALUES (741852963, null, 3)
210 INSERT INTO ATENDIMENTOS
211     VALUES (643791824, null, 4)
212 INSERT INTO ATENDIMENTOS
213     VALUES (65498234, null, 5)
214 INSERT INTO ATENDIMENTOS
215     VALUES (238423558, 5146137, 6)
216
217 SELECT * FROM ATENDIMENTOS

```

120 %

Resultados Mensagens

	RamalPrincipal	RamalSecundario	IdFuncionario
1	65498234	NULL	5
2	123456789	NULL	1
3	238423558	5146137	6
4	643791824	NULL	4
5	741852963	NULL	3
6	987654321	1123456	2

Em sequência, a função de busca descrita seria:

```
CREATE PROCEDURE BUSCA
(
    @NomeDoFunc VARCHAR(25) --Variável para busca do nome
)
AS
BEGIN
    SELECT NomeFuncionario, NomeSecao, RamalPrincipal, RamalSecundario
    FROM FUNCIONARIOS INNER JOIN ATENDIMENTOS
    ON FUNCIONARIOS.IdFuncionario = ATENDIMENTOS.IdFuncionario
    INNER JOIN SECAO
    ON FUNCIONARIOS.IdSecao = SECAO.IdSecao
    WHERE FUNCIONARIOS.IdDepto = SECAO.IdDepto AND FUNCIONARIOS.IdDivisao = SECAO.IdDivisao AND NomeFuncionario LIKE '%' + @NomeDoFunc + '%'
END;
```

Executando o procedimento:

16
17 EXEC BUSCA @NomeDoFunc = 'O'

131 %

Resultados Mensagens

	NomeFuncionario	NomeSecao	RamalPrincipal	RamalSecundario
1	Moreno	Secao Vettel	65498234	NULL
2	Ayrton	Secao Inicial	123456789	NULL
3	Roberto	Secao Gabriel	643791824	NULL
4	Fabio	Secao Fábio	987654321	1123456

16
17 EXEC BUSCA @NomeDoFunc = 'An'

131 %

Resultados Mensagens

	NomeFuncionario	NomeSecao	RamalPrincipal	RamalSecundario
1	Tania	Secao Alonso	238423558	5146137
2	Ana	Secao Marianita	741852963	NULL

O Código se encontra na tabela abaixo:

CÓDIGO BUSCA PELO NOME

```
CREATE TABLE ATENDIMENTOS
(
    RamalPrincipal INT PRIMARY KEY,
    RamalSecundario INT,
    IdFuncionario INT NOT NULL,
    CONSTRAINT Fk_IdFunc FOREIGN KEY (IdFuncionario) REFERENCES FUNCIONARIOS
(IdFuncionario),
);

INSERT INTO ATENDIMENTOS
VALUES (123456789, null, 1)
INSERT INTO ATENDIMENTOS
VALUES (987654321, 1123456, 2)
INSERT INTO ATENDIMENTOS
VALUES (741852963, null, 3)
INSERT INTO ATENDIMENTOS
VALUES (643791824, null, 4)
INSERT INTO ATENDIMENTOS
VALUES (65498234, null, 5)
INSERT INTO ATENDIMENTOS
VALUES (238423558, 5146137, 6)

SELECT * FROM ATENDIMENTOS

CREATE PROCEDURE BUSCA
(
    @NomeDoFunc VARCHAR(25) --Variável para busca do nome
)
AS
BEGIN

    SELECT NomeFuncionario, NomeSecao, RamalPrincipal, RamalSecundario
    FROM FUNCIONARIOS INNER JOIN ATENDIMENTOS
    ON FUNCIONARIOS.IdFuncionario = ATENDIMENTOS.IdFuncionario
    INNER JOIN SECAO
    ON FUNCIONARIOS.IdSecao = SECAO.IdSecao
    WHERE FUNCIONARIOS.IdDepto = SECAO.IdDepto AND
    FUNCIONARIOS.IdDivisao = SECAO.IdDivisao AND NomeFuncionario LIKE '%' +
    @NomeDoFunc + '%'

END;

EXEC BUSCA @NomeDoFunc = 'An'
```

6. Tomando por base as tabelas já definidas e utilizadas, criar um Trigger para gerar Log das operações de alteração de SALARIO. A nova tabela de log deverá conter além dos dados de identificação, inclusive data e hora, o salario antigo e o novo. A Trigger deverá consistir se o salário novo é menor que o Salario antigo e caso afirmativo não permitir a alteração.

Para resolução do Exercício, primeiro será necessário criar a tabela de LOGs:

```
CREATE TABLE LOGs
(
    IdFuncionario INT,
    DataAlteracaoSalario DATETIME NOT NULL,
    VlSalarioAntigo FLOAT NOT NULL,
    VlSalarioNovo FLOAT NOT NULL,
    PRIMARY KEY (IdFuncionario, DataAlteracaoSalario),
    CONSTRAINT Fk_LOG_ FOREIGN KEY (IdFuncionario) REFERENCES FUNCIONARIOS (IdFuncionario),
);
```

Em sequência, criando o TRIGGER para alterações nos salários dos funcionários:

```
30 CREATE TRIGGER AlteracaoSalario
31 ON FUNCIONARIOS
32 FOR UPDATE
33 AS
34 IF (SELECT VlSalario FROM INSERTED) < (SELECT VlSalario FROM DELETED)
35 BEGIN
36     PRINT '#####'
37     PRINT 'O Salário alterado não pode ser menor que o anterior!!!'
38     PRINT '#####'
39     ROLLBACK TRANSACTION
40 END
41
42 ELSE
43 IF (SELECT VlSalario FROM INSERTED) > (SELECT VlSalario FROM DELETED)
44 BEGIN
45     PRINT '#####'
46     PRINT 'O Salário será alterado com sucesso!'
47     PRINT '#####'
```

```

49  --Variáveis para armazenar os valores
50  DECLARE @IdFuc INT;
51  DECLARE @Data DATETIME;
52  DECLARE @VlAntigo FLOAT;
53  DECLARE @VlNovo FLOAT;
54
55  --Passando os valores
56  SELECT @IdFuc = i.IdFuncionario, @VlNovo = i.VlSalario
57  FROM inserted i;
58  SELECT @VlAntigo = d.VlSalario
59  FROM deleted d;
60  SET @Data = GETDATE();
61
62  --Passando para a tabela LOGs
63  INSERT INTO LOGs
64  VALUES(@IdFuc, @Data, @VlAntigo, @VlNovo)
65
66
67  --Selecionando a alteração do LOGs
68
69  SELECT * FROM LOGs
70
71  END

```

Por fim, abaixo temos a execução do TRIGGER:

A- Como os dados estavam antes das alterações:

72
73
74

```
SELECT * FROM FUNCIONARIOS
```

131 %

Resultados Mensagens

	IdFuncionario	NomeFuncionario	DtAdmissao	IdCargo	VlSalario	IdSecao	IdDivisao	IdDepto	DtUltFeria	DtDemissao	CPF
1	1	Ayrton	2022-09-03	111	5020,3	900	1	10	2023-10-18	NULL	6547824
2	2	Fabio	2022-09-03	222	3020,3	903	1	10	2023-01-09	NULL	254858
3	3	Ana	2021-11-03	333	2020,3	800	11	20	2022-10-09	2023-10-10	5421878
4	4	Roberto	2021-10-08	444	6054,3	700	101	30	2023-04-02	NULL	3167441
5	5	Moreno	2022-09-05	555	3333,3	600	201	40	2023-04-16	NULL	316040
6	6	Tania	2020-09-15	333	1012,3	601	201	40	2023-08-18	2023-10-10	324663

B- Realizando uma alteração que deve ocorrer com sucesso

```

75 UPDATE FUNCIONARIOS
76 SET VLSalario = 1050.3
77 WHERE IdFuncionario = 6

```

131 %

Resultados Mensagens

	IdFuncionario	DataAlteracaoSalario	VLSalarioAntigo	VLSalarioNovo
1	6	2023-10-19 12:11:23.880	1050,3	1050,3

```

72
73 SELECT * FROM FUNCIONARIOS

```

131 %

Resultados Mensagens

	IdFuncionario	NomeFuncionario	DtAdmissao	IdCargo	VLSalario	IdSecao	IdDivisao	IdDepto	DtUltFeria	DtDemissao	CPF
1	1	Ayrton	2022-09-03	111	5020,3	900	1	10	2023-10-18	NULL	6547824
2	2	Fabio	2022-09-03	222	3020,3	903	1	10	2023-01-09	NULL	254858
3	3	Ana	2021-11-03	333	2020,3	800	11	20	2022-10-09	2023-10-10	5421878
4	4	Roberto	2021-10-08	444	6054,3	700	101	30	2023-04-02	NULL	3167441
5	5	Moreno	2022-09-05	555	3333,3	600	201	40	2023-04-16	NULL	316040
6	6	Tania	2020-09-15	333	1050,3	601	201	40	2023-08-18	2023-10-10	324663

C- TRIGGER impedindo a alteração inválida:

```

74
75 UPDATE FUNCIONARIOS
76 SET VLSalario = 900.3
77 WHERE IdFuncionario = 6

```

131 %

Mensagens

```

#####
O Salário alterado não pode ser menor que o anterior!!!
#####
Mensagem 3609, Nível 16, Estado 1, Linha 75
A transação foi encerrada no gatilho. O lote foi anulado.

Horário de conclusão: 2023-10-19T12:12:33.8955998-03:00

```

```
SELECT * FROM FUNCIONARIOS
```

131 %

Resultados Mensagens

	IdFuncionario	NomeFuncionario	DtAdmissao	IdCargo	VlSalario	IdSecao	IdDivisao	IdDepto	DtUltFeria	DtDemissao	CPF
1	1	Ayrton	2022-09-03	111	5020,3	900	1	10	2023-10-18	NULL	6547824
2	2	Fabio	2022-09-03	222	3020,3	903	1	10	2023-01-09	NULL	254858
3	3	Ana	2021-11-03	333	2020,3	800	11	20	2022-10-09	2023-10-10	5421878
4	4	Roberto	2021-10-08	444	6054,3	700	101	30	2023-04-02	NULL	3167441
5	5	Moreno	2022-09-05	555	3333,3	600	201	40	2023-04-16	NULL	316040
6	6	Tania	2020-09-15	333	1050,3	601	201	40	2023-08-18	2023-10-10	324663

CÓDIGO LOGs E TRIGGERS

```
CREATE TRIGGER AlteracaoSalario
ON FUNCIONARIOS
FOR UPDATE
AS
IF (SELECT VlSalario FROM INSERTED) < (SELECT VlSalario FROM DELETED)
BEGIN
PRINT '#####'
PRINT 'O Salário alterado não pode ser menor que o anterior!!!'
PRINT '#####'
ROLLBACK TRANSACTION
END

ELSE
IF (SELECT VlSalario FROM INSERTED) > (SELECT VlSalario FROM
DELETED)
BEGIN
PRINT '#####'
PRINT 'O Salário será alterado com sucesso!'
PRINT '#####'

--Variáveis para armazenar os valores
DECLARE @IdFuc INT;
DECLARE @Data DATETIME;
DECLARE @VlAntigo FLOAT;
DECLARE @VlNovo FLOAT;

--Passando os valores
SELECT @IdFuc = i.IdFuncionario, @VlNovo = i.VlSalario
FROM inserted i;
SELECT @VlAntigo = d.VlSalario
FROM deleted d;
SET @Data = GETDATE();

--Passando para a tabela LOGs
INSERT INTO LOGs
VALUES(@IdFuc, @Data, @VlAntigo, @VlNovo)

--Selecionando a alteração do LOGs
```

```
SELECT * FROM LOGs

END

SELECT * FROM FUNCIONARIOS

UPDATE FUNCIONARIOS
SET VlSalario = 1060.3
WHERE IdFuncionario = 6
```

7. Criar um Trigger para que quando seja demitido um funcionário, seja também atualizada a tabela de FUNC_PROJ (Alocações) no campo dt_aloc_fim do respectivo funcionário.

Criando o TRIGGER:

```
79
80 CREATE TRIGGER DEMISSAO
81 ON FUNCIONARIOS
82 FOR UPDATE
83 AS
84 IF (SELECT DtDemissao FROM INSERTED) IS NOT NULL
85 BEGIN
86 DECLARE @DtDemissao DATE;
87 DECLARE @IdFun INT;
88
89 SELECT @DtDemissao = i.DtDemissao, @IdFun = i.IdFuncionario
90 FROM inserted i;
91
92 UPDATE FUNC_PROJ
93 SET DtAlocFim = @DtDemissao
94 WHERE FUNC_PROJ.IdFuncionario = @IdFun
95
96 END
```

Tabela FUNC_PROJ antes de um update:

```
98 SELECT * FROM FUNC_PROJ
99
```

108 %

	CodProj	IdFuncionario	DtAlocIni	DtAlocFim
1	10	1	2023-10-10	NULL
2	10	2	2023-10-10	NULL
3	20	3	2023-10-10	NULL
4	20	4	2023-10-10	NULL
5	30	5	2023-10-10	NULL
6	30	6	2023-10-10	NULL

```
99 SELECT * FROM FUNCIONARIOS
100 SELECT * FROM FUNC_PROJ
```

131 %

	IdFuncionario	NomeFuncionario	DtAdmissao	IdCargo	VSalarario	IdSecao	IdDivisao	IdDepto	DtUltFeria	DtDemissao	CPF
1	1	Ayrton	2022-09-03	111	5020,3	900	1	10	2023-10-18	NULL	6547824
2	2	Fabio	2022-09-03	222	3020,3	903	1	10	2023-01-09	NULL	254858
3	3	Ana	2021-11-03	333	2020,3	800	11	20	2022-10-09	2023-10-10	5421878
4	4	Roberto	2021-10-08	444	6054,3	700	101	30	2023-04-02	NULL	3167441
5	5	Moreno	2022-09-05	555	3333,3	600	201	40	2023-04-16	NULL	316040
6	6	Tania	2023-11-12	333	1060,3	601	201	40	2023-08-18	2023-10-10	324663

Realizando o update e testando o trigger:

```
104 UPDATE FUNCIONARIOS
105 SET DtDemissao = '2023-10-10'
106 WHERE IdFuncionario = 5
107
108 SELECT * FROM FUNCIONARIOS
109 SELECT * FROM FUNC_PROJ
```

9 %

	IdFuncionario	NomeFuncionario	DtAdmissao	IdCargo	VSalarario	IdSecao	IdDivisao	IdDepto	DtUltFeria	DtDemissao	CPF
1	1	Ayrton	2022-09-03	111	5020,3	900	1	10	2023-10-18	NULL	6547824
2	2	Fabio	2022-09-03	222	3020,3	903	1	10	2023-01-09	NULL	254858
3	3	Ana	2021-11-03	333	2020,3	800	11	20	2022-10-09	2023-10-10	5421878
4	4	Roberto	2021-10-08	444	6054,3	700	101	30	2023-04-02	NULL	3167441
5	5	Moreno	2023-10-10	555	3333,3	600	201	40	2023-04-16	2023-10-10	316040
6	6	Tania	2023-11-12	333	1060,3	601	201	40	2023-08-18	2023-10-10	324663

	CodProj	IdFuncionario	DtAlocIni	DtAlocFim
1	10	1	2023-10-10	NULL
2	10	2	2023-10-10	NULL
3	20	3	2023-10-10	2023-10-10
4	20	4	2023-10-10	NULL
5	30	5	2023-10-10	2023-10-10
6	30	6	2023-10-10	2023-10-10

CÓDIGO TRIGGER DEMISSÃO

```
CREATE TRIGGER DEMISSAO
ON FUNCIONARIOS
FOR UPDATE
AS
IF (SELECT DtDemissao FROM INSERTED) IS NOT NULL
BEGIN
    DECLARE @DtDemissao DATE;
    DECLARE @IdFun INT;

    SELECT @DtDemissao = i.DtDemissao, @IdFun = i.IdFuncionario
    FROM inserted i;

    UPDATE FUNC_PROJ
    SET DtAlocFim = @DtDemissao
    WHERE FUNC_PROJ.IdFuncionario = @IdFun
END

SELECT * FROM FUNCIONARIOS
SELECT * FROM FUNC_PROJ

UPDATE FUNCIONARIOS
SET DtDemissao = '2023-10-10'
WHERE IdFuncionario = 5

SELECT * FROM FUNCIONARIOS
SELECT * FROM FUNC_PROJ
```

8. Adaptar as Tabelas para que o Funcionário possa ser alocado em mais de um intervalo de datas ao mesmo Projeto. Criar Trigger para garantir a integridade das informações efetuando as consistências necessárias por ocasião de inclusões e alterações.

Atualizando a tabela FUNC_PROJ para que o funcionário possa ser alocado em até 2 intervalos de tempo:


```

1 ALTER TABLE FUNC_PROJ
2 ADD DataFimAlocInicial DATE,
3     DataIniSegundaAloc DATE;
4
5 SELECT * FROM FUNC_PROJ

```

108 %

Resultados Mensagens

	CodProj	IdFuncionario	DtAlocIni	DtAlocFim	DataFimAlocInicial	DataIniSegundaAloc
1	10	1	2023-10-10	NULL	NULL	NULL
2	10	2	2023-10-10	NULL	NULL	NULL
3	20	3	2023-10-10	2023-10-10	NULL	NULL
4	20	4	2023-10-10	NULL	NULL	NULL
5	30	5	2023-10-10	2023-10-10	NULL	NULL
6	30	6	2023-10-10	2023-10-10	NULL	NULL

Criando o TRIGGER:

A) Declarando e atribuindo os valores das variáveis

```

CREATE TRIGGER ALTERACOES_ALOCACAO
ON FUNC_PROJ
FOR UPDATE, INSERT
AS
    DECLARE @DtAlocFinal DATE
    DECLARE @DtAlocInicial DATE
    DECLARE @DtFimAlocInicial DATE --ARMAZENA FIM DO 1º INTERVALOR
    DECLARE @DtInicioAlocSegunda DATE --ARMAZENA INICIO DO 2º INTERVALO
    DECLARE @DtAdmissao DATE
    DECLARE @DtDemissao DATE
    DECLARE @DtInicioProjeto DATE
    DECLARE @DtFimProjeto DATE

```

```

SELECT
@DtAlocInicial = i.DtAlocIni,
@DtAlocFinal = i.DtAlocFim,
@DtFimAlocInicial = i.DataFimAlocInicial,
@DtInicioAlocSegunda = i.DataIniSegundaAloc
FROM inserted i;

SELECT @DtAdmissao = DtAdmissao, @DtDemissao = DtDemissao
FROM FUNCIONARIOS INNER JOIN inserted
ON FUNCIONARIOS.IdFuncionario = inserted.IdFuncionario

SELECT @DtInicioProjeto = DtInicio, @DtFimProjeto = DtFim
FROM PROJETO INNER JOIN inserted
ON inserted.CodProj = PROJETO.CodProj

```

Iniciando as sequências de segurança das alterações:

```

--Iniciando sequencia de IFs Para integridade
--Caso 1: DtAlocIni Deve ser maior ou igual DtAdmissão

IF @DtAlocInicial < @DtAdmissao
BEGIN
PRINT '#####';
PRINT 'ERRO : A DATA DE ALOCAÇÃO INICIAL DEVE SER AO MENOS IGUAL A DATA DE ADMISSÃO!!!';
PRINT 'CANCELANDO TRANSAÇÃO'
PRINT '#####';
ROLLBACK TRANSACTION
END

--Caso 2: DtAlocFim deve ser no máximo igual DtDemissão

ELSE
IF (@DtAlocFinal > @DtDemissao AND @DtDemissao IS NOT NULL)
BEGIN
PRINT '#####';
PRINT 'ERRO : A DATA DE FIM DE ALOCAÇÃO DEVE SER MENOR OU IGUAL A DATA DE DEMISSÃO!!!';
PRINT 'CANCELANDO TRANSAÇÃO'
PRINT '#####';
ROLLBACK TRANSACTION
END

```

OBS -> Os próximos códigos podem ser melhor visualizados na tabela
'CÓDIGO TRIGGER ALTERAÇÃO DATAS'

--Caso 3: DtAlocInicial deve ser a menor data

```
ELSE
IF (@DtAlocInicial > @DtFimAlocInicial) OR (@DtAlocInicial > @DtInicioAlocSegunda) OR (@DtAlocInicial > @DtAlocFinal)
BEGIN
PRINT '#####';
PRINT 'ERRO : A DE ALOCAÇÃO INICIAL DEVE SER A MENOR DATA!!!!!!';
PRINT 'CANCELANDO TRANSAÇÃO'
PRINT '#####';
ROLLBACK TRANSACTION
END
```

--Caso 4: DtFimAlocInicial deve ser maior que DtAlocInicial e menor que
--DtInicioAlocSegunda e DtAlocFinal, DESDE QUE não seja NULO
--(2 intervalos de operação no projeto)

```
ELSE
IF( (@DtFimAlocInicial < @DtAlocInicial) OR (@DtFimAlocInicial > @DtInicioAlocSegunda) OR (@DtFimAlocInicial > @DtAlocFinal) ) AND @DtFimAlocInicial IS NOT NULL
BEGIN
PRINT '#####';
PRINT 'ERRO : A DATA DE FIM DA 1º ALOCAÇÃO DEVE SER MAIOR QUE A DATA DE ALOCAÇÃO INICIAL'
PRINT 'E MENOR QUE AS DATAS DE INICIO DA SEGUNDA ALOCAÇÃO E FIM DA ALOCAÇÃO!!!'
PRINT 'CANCELANDO TRANSAÇÃO'
PRINT '#####';
ROLLBACK TRANSACTION
END
```

--Caso 5: DtInicioAlocSegunda deve ser maior que DtAlocInicial e DtFimAlocInicial,
--e menor que a DtAlocFinal, DESDE QUE não seja nulo
--(2 intervalos de operação no projeto)

```
ELSE
IF( (@DtInicioAlocSegunda < @DtAlocInicial) OR (@DtInicioAlocSegunda < @DtFimAlocInicial) OR (@DtInicioAlocSegunda > @DtAlocFinal) ) AND @DtInicioAlocSegunda IS NOT NULL
BEGIN
PRINT '#####';
PRINT 'ERRO : A DATA DE INÍCIO DA 2º ALOCAÇÃO DEVE SER MAIOR QUE A DATA DE ALOCAÇÃO INICIAL'
PRINT 'E FIM DA 1º ALOCAÇÃO, ALÉM DE SER MENOR QUE O FIM DA ALOCAÇÃO TOTAL!!!'
PRINT 'CANCELANDO TRANSAÇÃO'
PRINT '#####';
ROLLBACK TRANSACTION
END
```

--Caso 6: DtFimAloc deve ser a maior data SE NÃO NULO

```
ELSE
IF( (@DtAlocFinal < @DtAlocInicial) OR (@DtAlocFinal < @DtFimAlocInicial) OR (@DtAlocFinal < @DtInicioAlocSegunda) ) AND @DtAlocFinal IS NOT NULL
BEGIN
PRINT '#####';
PRINT 'ERRO : A DATA DE FIM DE ALOCAÇÃO DEVE SER A MAIOR DO PROJETO!!!!'
PRINT 'CANCELANDO TRANSAÇÃO'
PRINT '#####';
ROLLBACK TRANSACTION
END
```

--Caso 7: Todas as datas deve ser maiores que a de inicio do projeto, apenas a DtAlocInicial pode ser IGUAL

```
ELSE
IF( (@DtAlocInicial < @DtInicioProjeto) OR (@DtFimAlocInicial <= @DtInicioProjeto) OR (@DtInicioAlocSegunda <= @DtInicioProjeto) OR ( @DtAlocFinal <= @DtInicioProjeto) )
BEGIN
PRINT '#####';
PRINT 'ERRO : A DATA DE INICIO DO PROJETO É MAIOR OU IGUAL QUE ALGUMA DAS DATAS!!!!'
PRINT 'CANCELANDO TRANSAÇÃO'
PRINT '#####';
ROLLBACK TRANSACTION
END
```

--Caso 8: Todas as datas devem ser menores que a de fim do projeto, apenas DtAlocFinal pode ser IGUAL

```
ELSE
IF( (@DtAlocInicial >= @DtFimProjeto) OR (@DtFimAlocInicial >= @DtFimProjeto) OR (@DtInicioAlocSegunda >= @DtFimProjeto) OR ( @DtAlocFinal > @DtFimProjeto) ) AND @DtFimProjeto IS NOT NULL
BEGIN
PRINT '#####';
PRINT 'ERRO : A DATA DE FIM DO PROJETO É MENOR OU IGUAL QUE ALGUMA DAS DATAS!!!!'
PRINT 'CANCELANDO TRANSAÇÃO'
PRINT '#####';
ROLLBACK TRANSACTION
END
```

```
ELSE
SELECT * FROM FUNC_PROJ
```

B) Visualizando as tabelas de FUNCIONÁRIOS, PROJETOS e FUN_PROJ inicialmente (Durante seus inserts, NÃO HOVERAM AS PREOCUPAÇÕES RELATIVAS AOS CASOS TRATADOS PELO TRIGGER, por isso alguns dados podem ser conflitantes)

	IdFuncionario	NomeFuncionario	DtAdmissao	IdCargo	VSalario	IdSecao	IdDivisao	IdDepto	DtUltFeria	DtDemissao	CPF
1	1	Ayrton	2022-09-03	111	5020,3	900	1	10	2023-10-18	NULL	6547824
2	2	Fabio	2022-09-03	222	3020,3	903	1	10	2023-01-09	NULL	254858
3	3	Ana	2021-11-03	333	2020,3	800	11	20	2022-10-09	2023-10-10	5421878
4	4	Roberto	2021-10-08	444	6054,3	700	101	30	2023-04-02	NULL	3167441
5	5	Moreno	2023-10-10	555	3333,3	600	201	40	2023-04-16	2023-10-10	316040
6	6	Tania	2023-11-12	333	1060,3	601	201	40	2023-08-18	2023-10-10	324663

	CodProj	IdFuncionario	DtAlocIni	DtAlocFim	DataFimAlocInicial	DataIniSegundaAloc
1	10	1	2023-10-10	2023-10-30	2023-10-15	2023-10-20
2	10	2	2023-10-10	NULL	NULL	NULL
3	20	3	2023-10-10	2023-10-10	NULL	NULL
4	20	4	2023-10-10	NULL	NULL	NULL
5	30	5	2023-10-10	2023-10-10	NULL	NULL
6	30	6	2023-10-10	2023-10-10	NULL	NULL

	CodProj	SiglaProj	DtInicio	DtFim	StatusProj
1	10	ContratacaoAnual	2023-08-17	NULL	ATIVO
2	20	RelatorioMensal	2023-10-10	2023-10-18	COMPLETO
3	30	SistemaPIX	2023-09-17	NULL	ATIVO

Inserindo data de demissão para Funcionário 1:

Resultados Mensagens											
	IdFuncionario	NomeFuncionario	DtAdmissao	IdCargo	VSalario	IdSecao	IdDivisao	IdDepto	DtUltFeria	DtDemissao	CPF
1	1	Ayrton	2022-09-03	111	5020,3	900	1	10	2023-10-18	2023-12-12	6547824
2	2	Fabio	2022-09-03	222	3020,3	903	1	10	2023-01-09	NULL	254858
3	3	Ana	2021-11-03	333	2020,3	800	11	20	2022-10-09	2023-10-10	5421878
4	4	Roberto	2021-10-08	444	6054,3	700	101	30	2023-04-02	NULL	3167441
5	5	Moreno	2023-10-10	555	3333,3	600	201	40	2023-04-16	2023-10-10	316040
6	6	Tania	2023-11-12	333	1060,3	601	201	40	2023-08-18	2023-10-10	324663

C) Realizando um update bem-sucedido:

```

153 UPDATE FUNC_PROJ
154 SET DataFimAlocInicial = '12/10/2023', DataIniSegundaAloc = '22/10/2023', DtAlocFim = '29/10/2023'
155 WHERE IdFuncionario = 1
156

```

CodProj	IdFuncionario	DtAlocIni	DtAlocFim	DataFimAlocInicial	DataIniSegundaAloc
10	1	2023-10-10	2023-10-29	2023-10-12	2023-10-22

D) Provocando o erro 1 “DtAlocIni Deve ser maior ou igual DtAdmissão”

```

162 --TESTE 2: PROVOCANDO ERRO 1 -> DtAlocIni Deve ser maior ou igual DtAdmissão
163 UPDATE FUNC_PROJ
164 SET DtAlocIni = '10/11/2014'
165 WHERE IdFuncionario = 1
166 SELECT * FROM FUNC_PROJ

```

ERRO : A DATA DE ALOCAÇÃO INICIAL DEVE SER AO MENOS IGUAL A DATA DE ADMISSÃO!!!
 CANCELANDO TRANSAÇÃO
 Mensagem 3609, Nível 16, Estado 1, Linha 163
 A transação foi encerrada no gatilho. O lote foi anulado.
 Horário de conclusão: 2023-10-19T15:46:22.7059672-03:00

Alteração não foi realizada:

```

166 SELECT * FROM FUNC_PROJ
167

```

	CodProj	IdFuncionario	DtAlocIni	DtAlocFim	DataFimAlocInicial	DataIniSegundaAloc
1	10	1	2023-10-10	2023-10-29	2023-10-12	2023-10-22
2	10	2	2023-10-10	NULL	NULL	NULL
3	20	3	2023-10-10	2023-10-10	NULL	NULL
4	20	4	2023-10-10	NULL	NULL	NULL
5	30	5	2023-10-10	2023-10-10	NULL	NULL
6	30	6	2023-10-10	2023-10-10	NULL	NULL

E) Provocando erro 2 'DtAlocFim deve ser no máximo igual DtDemissão'

Primeiro, adicionando data de demissão:

```

170 UPDATE FUNCIONARIOS
171     SET DtDemissao = '12/12/2023'
172     WHERE IdFuncionario = 1
173 SELECT * FROM FUNCIONARIOS

```

113 %

Resultados Mensagens

	IdFuncionario	NomeFuncionario	DtAdmissao	IdCargo	VISalario	IdSecao	IdDivisao	IdDepto	DtUltFeria	DtDemissao	CPF
1	1	Ayrton	2022-09-03	111	5020,3	900	1	10	2023-10-18	2023-12-12	6547824
2	2	Fabio	2022-09-03	222	3020,3	903	1	10	2023-01-09	NULL	254858
3	3	Ana	2021-11-03	333	2020,3	800	11	20	2022-10-09	2023-10-10	5421878
4	4	Roberto	2021-10-08	444	6054,3	700	101	30	2023-04-02	NULL	3167441
5	5	Moreno	2023-10-10	555	3333,3	600	201	40	2023-04-16	2023-10-10	316040
6	6	Tania	2023-11-12	333	1060,3	601	201	40	2023-08-18	2023-10-10	324663

Provocando a falha:

```

178 UPDATE FUNC_PROJ
179     SET DtAlocFim = '30/12/2023'
180     WHERE IdFuncionario = 1

```

13 %

Mensagens

```

#####
ERRO : A DATA DE FIM DE ALOCAÇÃO DEVE SER MENOR OU IGUAL A DATA DE DEMISSÃO!!!
CANCELANDO TRANSAÇÃO
#####
Mensagem 3609, Nível 16, Estado 1, Linha 178
A transação foi encerrada no gatilho. O lote foi anulado.

```


Sem alterações realizadas:

F) Provocando erro 3 'DtAlocInicial deve ser a menor data'

183
184 --TESTE 4: PROVOCANDO ERRO 3 -> DtAlocInicial deve ser a menor data
185 UPDATE FUNC_PROJ
186 SET DtAlocIni='12/12/2023', DtAlocFim = '02/12/2023'
187 WHERE IdFuncionario = 1
188 SELECT * FROM FUNC_PROJ

13 %

Mensagens

```
#####
ERRO : A DE ALOCAÇÃO INICIAL DEVE SER A MENOR DATA!!!!!!
CANCELANDO TRANSAÇÃO
#####
Mensagem 3609, Nivel 16, Estado 1, Linha 185
A transação foi encerrada no gatilho. O lote foi anulado.
```

	2	10	2	2023-10-10	NULL	NULL	NULL
3	20	3	2023-10-10	2023-10-10	NULL	NULL	NULL
4	20	4	2023-10-10	NULL	NULL	NULL	NULL
5	30	5	2023-10-10	2023-10-10	NULL	NULL	NULL
6	30	6	2023-10-10	2023-10-10	NULL	NULL	NULL

Sem alterações realizadas:

188 SELECT * FROM FUNC_PROJ

113 %

Resultados Mensagens

	CodProj	IdFuncionario	DtAlocIni	DtAlocFim	DataFimAlocInicial	DataIniSegundaAloc
1	10	1	2023-10-10	2023-12-12	2023-10-12	2023-10-22
2	10	2	2023-10-10	NULL	NULL	NULL
3	20	3	2023-10-10	2023-10-10	NULL	NULL
4	20	4	2023-10-10	NULL	NULL	NULL
5	30	5	2023-10-10	2023-10-10	NULL	NULL
6	30	6	2023-10-10	2023-10-10	NULL	NULL

G) Provocando erro 4, segurança da data final da 1º alocação:

```

191 --TESTE 5: PROVOCANDO ERRO 4 -> DtFimAlocInicial deve ser maior que DtAlocInicial e menor que
192 --DtInicioAlocSegunda e DtAlocFinal, DESDE QUE não seja NULO
193 --(2 intervalos de operação no projeto)
194 UPDATE FUNC_PROJ
195 SET DtAlocIni='01/12/2023', DataFimAlocInicial = '09/12/2023', DataIniSegundaAloc = '08/12/2023', DtAlocFim = '11/12/2023'
196 WHERE IdFuncionario = 1
197 SELECT * FROM FUNC_PROJ
  
```

Mensagens

```

#####
ERRO : A DATA DE FIM DA 1ª ALOCAÇÃO DEVE SER MAIOR QUE A DATA DE ALOCAÇÃO INICIAL
E MENOR QUE AS DATAS DE INICIO DA SEGUNDA ALOCAÇÃO E FIM DA ALOCAÇÃO!!!
CANCELANDO TRANSAÇÃO
#####
Mensagem 3609, Nível 16, Estado 1, Linha 194
A transação foi encerrada no gatilho. O lote foi anulado.
  
```

Sem alterações feitas:

196 WHERE IdFuncionario = 1
197 SELECT * FROM FUNC_PROJ

113 %

	CodProj	IdFuncionario	DtAlocIni	DtAlocFim	DataFimAlocInicial	DataIniSegundaAloc
1	10	1	2023-10-10	2023-12-12	2023-10-12	2023-10-22
2	10	2	2023-10-10	NULL	NULL	NULL
3	20	3	2023-10-10	2023-10-10	NULL	NULL
4	20	4	2023-10-10	NULL	NULL	NULL
5	30	5	2023-10-10	2023-10-10	NULL	NULL
6	30	6	2023-10-10	2023-10-10	NULL	NULL

H) Provocando erro 5 -> Segurança da DtInicioAlocSegunda

```

200 --TESTE 6: PROVOCANDO ERRO 5 -> DtInicioAlocSegunda deve ser maior que DtAlocInicial e DtFimAlocInicial,
201 --e menor que a DtAlocFinal, DESDE QUE não seja nulo
202 --(2 intervalos de operação no projeto)
203 UPDATE FUNC_PROJ
204 SET DtAlocIni='01/12/2023', DataFimAlocInicial = '04/12/2023', DataIniSegundaAloc = '07/12/2023', DtAlocFim = '06/12/2023'
205 WHERE IdFuncionario = 1
  
```

Mensagens

```

#####
ERRO : A DATA DE INÍCIO DA 2ª ALOCAÇÃO DEVE SER MAIOR QUE A DATA DE ALOCAÇÃO INICIAL
E FIM DA 1ª ALOCAÇÃO, ALÉM DE SER MENOR QUE O FIM DA ALOCAÇÃO TOTAL!!!
CANCELANDO TRANSAÇÃO
#####
Mensagem 3609, Nível 16, Estado 1, Linha 203
A transação foi encerrada no gatilho. O lote foi anulado.
  
```


Sem alterações:

I) Provocando erro 6 – DtFimAloc deve ser a maior data SE NÃO NULO

```

209 --TESTE 7: PROVOCANDO ERRO 6 -> DtFimAloc deve ser a maior data SE NÃO NULO
210 UPDATE FUNC_PROJ
211 SET DtAlocIni='01/12/2023', DataFimAlocInicial = '04/12/2023', DataIniSegundaAloc = '07/12/2023', DtAlocFim = '02/12/2023'
212 WHERE IdFuncionario = 1

```

Mensagens

```

#####
ERRO : A DATA DE FIM DA 1ª ALOCAÇÃO DEVE SER MAIOR QUE A DATA DE ALOCAÇÃO INICIAL
E MENOR QUE AS DATAS DE INICIO DA SEGUNDA ALOCAÇÃO E FIM DA ALOCAÇÃO!!!
CANCELANDO TRANSAÇÃO
#####
Mensagem 3609, Nível 16, Estado 1, Linha 210
A transação foi encerrada no gatilho. O lote foi anulado.

```

206 SELECT * FROM FUNC_PROJ

	CodProj	IdFuncionario	DtAlocIni	DtAlocFim	DataFimAlocInicial	DataIniSegundaAloc
1	10	1	2023-10-10	2023-10-29	2023-10-12	2023-10-22
2	10	2	2023-10-10	NULL	NULL	NULL
3	20	3	2023-10-10	2023-10-10	NULL	NULL
4	20	4	2023-10-10	NULL	NULL	NULL
5	30	5	2023-10-10	2023-10-10	NULL	NULL
6	30	6	2023-10-10	2023-10-10	NULL	NULL

NOTA -> O erro 6 é simplesmente impossível de provocar, foi criado como redundância de segurança. Os Ifs anterior protegem o BD da falha 6.

J) Provocando erro 7 -> Todas as datas deve ser maiores que a de inicio do projeto, apenas a DtAlocInicial pode ser IGUAL

```

215 --TESTE 8: PROVOCANDO ERRO 7 -> Todas as datas deve ser maiores que a de inicio do projeto, apenas a DtAlocInicial pode ser IGUAL
216 UPDATE FUNC_PROJ
217 SET DtAlocIni='17/07/2023', DataFimAlocInicial = '04/12/2023', DataIniSegundaAloc = '07/12/2023', DtAlocFim = '12/12/2023'
218 WHERE IdFuncionario = 1
219

```

Mensagens

```

#####
ERRO : A DATA DE INICIO DO PROJETO É MAIOR OU IGUAL QUE ALGUMA DAS DATAS!!!!
CANCELANDO TRANSAÇÃO
#####
Mensagem 3609, Nível 16, Estado 1, Linha 216
A transação foi encerrada no gatilho. O lote foi anulado.

```

Sem alterações

K) Provocando erro 8 -> Todas as datas devem ser menores que a de fim do projeto, apenas DtAlocFinal pode ser IGUAL

```

221 --TESTE 9: PROVOCANDO ERRO 8 -> Todas as datas devem ser menores que a de fim do projeto, apenas DtAlocFinal pode ser IGUAL
222 --Inserindo data final no projeto
223 UPDATE PROJETO
224     SET DtFim = '10/12/2023'
225     WHERE CodProj = 10
226
227 UPDATE FUNC_PROJ
228     SET DtAlocIni='02/12/2023', DataFimAlocInicial = '04/12/2023', DataIniSegundaAloc = '07/12/2023', DtAlocFim = '12/12/2023'
229     WHERE IdFuncionario = 1
  
```

Mensagens

```

#####
ERRO : A DATA DE FIM DO PROJETO É MENOR OU IGUAL QUE ALGUMA DAS DATAS!!!!
CANCELANDO TRANSAÇÃO
#####
Mensagem 3609, Nivel 16, Estado 1, Linha 227
A transação foi encerrada no gatilho. O lote foi anulado.
  
```

5	30	5	2023-10-10	2023-10-10	NULL	NULL
6	30	6	2023-10-10	2023-10-10	NULL	NULL

Sem alterações:

230 SELECT * FROM FUNC_PROJ

231

113 %

	CodProj	IdFuncionario	DtAlocIni	DtAlocFim	DataFimAlocInicial	DataIniSegundaAloc
1	10	1	2023-10-10	2023-10-29	2023-10-12	2023-10-22
2	10	2	2023-10-10	NULL	NULL	NULL
3	20	3	2023-10-10	2023-10-10	NULL	NULL
4	20	4	2023-10-10	NULL	NULL	NULL
5	30	5	2023-10-10	2023-10-10	NULL	NULL
6	30	6	2023-10-10	2023-10-10	NULL	NULL

O código pode ser copiado:

TABELA TRIGGER DATAS
<pre> ALTER TABLE FUNC_PROJ ADD DataFimAlocInicial DATE, DataIniSegundaAloc DATE; </pre>

```

SELECT * FROM FUNC_PROJ

CREATE TRIGGER ALTERACOES_ALOCACAO
ON FUNC_PROJ
FOR UPDATE, INSERT
AS
    DECLARE @DtAlocFinal DATE
    DECLARE @DtAlocInicial DATE
    DECLARE @DtFimAlocInicial DATE --ARMAZENA FIM DO 1º INTERVALOR
    DECLARE @DtInicioAlocSegunda DATE --ARMAZENA INICIO DO 2º INTERVALO
    DECLARE @DtAdmissao DATE
    DECLARE @DtDemissao DATE
    DECLARE @DtInicioProjeto DATE
    DECLARE @DtFimProjeto DATE

    SELECT
        @DtAlocInicial = i.DtAlocIni,
        @DtAlocFinal = i.DtAlocFim,
        @DtFimAlocInicial = i.DataFimAlocInicial,
        @DtInicioAlocSegunda = i.DataIniSegundaAloc
    FROM inserted i;

    SELECT @DtAdmissao = DtAdmissao, @DtDemissao = DtDemissao
    FROM FUNCIONARIOS INNER JOIN inserted
    ON FUNCIONARIOS.IdFuncionario = inserted.IdFuncionario

    SELECT @DtInicioProjeto = DtInicio, @DtFimProjeto = DtFim
    FROM PROJETO INNER JOIN inserted
    ON inserted.CodProj = PROJETO.CodProj

    --Iniciando sequencia de IFs Para integridade
    --Caso 1: DtAlocIni Deve ser maior ou igual DtAdmissão

    IF @DtAlocInicial < @DtAdmissao
        BEGIN
            PRINT
            '#####'
            ;
            PRINT 'ERRO : A DATA DE ALOCAÇÃO INICIAL DEVE SER AO MENOS IGUAL A
            DATA DE ADMISSÃO!!!';
            PRINT 'CANCELANDO TRANSAÇÃO'
            PRINT
            '#####'
            ;
            ROLLBACK TRANSACTION
        END

    --Caso 2: DtAlocFim deve ser no máximo igual DtDemissão

    ELSE
    IF (@DtAlocFinal > @DtDemissao AND @DtDemissao IS NOT NULL)
        BEGIN
            PRINT
            '#####'
            ;
            PRINT 'ERRO : A DATA DE FIM DE ALOCAÇÃO DEVE SER MENOR OU IGUAL A
            DATA DE DEMISSÃO!!!'
            PRINT 'CANCELANDO TRANSAÇÃO'
            PRINT
            '#####'

```

```

;
    ROLLBACK TRANSACTION
END

--Caso 3: DtAlocInicial deve ser a menor data

ELSE
    IF (@DtAlocInicial > @DtFimAlocInicial) OR (@DtAlocInicial >
@DtInicioAlocSegunda) OR (@DtAlocInicial > @DtAlocFinal)
        BEGIN
            PRINT
            '#####'
;
            PRINT 'ERRO : A DE ALOCAÇÃO INICIAL DEVE SER A MENOR DATA!!!!!!'
            PRINT 'CANCELANDO TRANSAÇÃO'
            PRINT
            '#####'
;
            ROLLBACK TRANSACTION
        END

--Caso 4: DtFimAlocInicial deve ser maior que DtAlocInicial e menor que
--DtInicioAlocSegunda e DtAlocFinal, DESDE QUE não seja NULO
--(2 intervalos de operação no projeto)

ELSE
    IF( (@DtFimAlocInicial < @DtAlocInicial) OR (@DtFimAlocInicial >
@DtInicioAlocSegunda) OR (@DtFimAlocInicial > @DtAlocFinal) ) AND
@DtFimAlocInicial IS NOT NULL
        BEGIN
            PRINT
            '#####'
;
            PRINT 'ERRO : A DATA DE FIM DA 1º ALOCAÇÃO DEVE SER MAIOR QUE A
DATA DE ALOCAÇÃO INICIAL'
            PRINT 'E MENOR QUE AS DATAS DE INICIO DA SEGUNDA ALOCAÇÃO E FIM DA
ALOCAÇÃO!!!'
            PRINT 'CANCELANDO TRANSAÇÃO'
            PRINT
            '#####'
;
            ROLLBACK TRANSACTION
        END

--Caso 5: DtInicioAlocSegunda deve ser maior que DtAlocInicial e
DtFimAlocInicial,
--e menor que a DtAlocFinal, DESDE QUE não seja nulo
--(2 intervalos de operação no projeto)

ELSE
    IF( (@DtInicioAlocSegunda < @DtAlocInicial) OR (@DtInicioAlocSegunda <
@DtFimAlocInicial) OR (@DtInicioAlocSegunda > @DtAlocFinal) ) AND
@DtInicioAlocSegunda IS NOT NULL
        BEGIN
            PRINT
            '#####'
;
            PRINT 'ERRO : A DATA DE INÍCIO DA 2º ALOCAÇÃO DEVE SER MAIOR QUE A
DATA DE ALOCAÇÃO INICIAL'
            PRINT 'E FIM DA 1º ALOCAÇÃO, ALÉM DE SER MENOR QUE O FIM DA
ALOCAÇÃO TOTAL!!!'

```

```

        PRINT 'CANCELANDO TRANSAÇÃO'
        PRINT
        '#####'
;
        ROLLBACK TRANSACTION
    END

    --Caso 6: DtFimAloc deve ser a maior data SE NÃO NULO

    ELSE
        IF( (@DtAlocFinal < @DtAlocInicial) OR (@DtAlocFinal < @DtFimAlocInicial)
OR (@DtAlocFinal < @DtInicioAlocSegunda) ) AND @DtAlocFinal IS NOT NULL
        BEGIN
            PRINT
            '#####'
;
            PRINT 'ERRO : A DATA DE FIM DE ALOCAÇÃO DEVE SER A MAIOR DO
PROJETO!!!!'
            PRINT 'CANCELANDO TRANSAÇÃO'
            PRINT
            '#####'
;
            ROLLBACK TRANSACTION
        END

        --Caso 7: Todas as datas deve ser maiores que a de inicio do projeto,
        apenas a DtAlocInicial pode ser IGUAL

        ELSE
            IF( (@DtAlocInicial < @DtInicioProjeto) OR (@DtFimAlocInicial <=
@DtInicioProjeto) OR (@DtInicioAlocSegunda <= @DtInicioProjeto) OR ( @DtAlocFinal
<= @DtInicioProjeto) )
            BEGIN
                PRINT
                '#####'
;
                PRINT 'ERRO : A DATA DE INICIO DO PROJETO É MAIOR OU IGUAL QUE
ALGUMA DAS DATAS!!!!'
                PRINT 'CANCELANDO TRANSAÇÃO'
                PRINT
                '#####'
;
                ROLLBACK TRANSACTION
            END

            --Caso 8: Todas as datas devem ser menores que a de fim do projeto, apenas
            DtAlocFinal pode ser IGUAL

            ELSE
                IF( (@DtAlocInicial >= @DtFimProjeto) OR (@DtFimAlocInicial >=
@DtFimProjeto) OR (@DtInicioAlocSegunda >= @DtFimProjeto) OR ( @DtAlocFinal >
@DtFimProjeto) ) AND @DtFimProjeto IS NOT NULL
                BEGIN
                    PRINT
                    '#####'
;
                    PRINT 'ERRO : A DATA DE FIM DO PROJETO É MENOR OU IGUAL QUE ALGUMA
DAS DATAS!!!!'
                    PRINT 'CANCELANDO TRANSAÇÃO'
                    PRINT
                    '#####'

```

```
;
    ROLLBACK TRANSACTION
END

ELSE
SELECT * FROM FUNC_PROJ

--TESTANDO VALORES

--TESTE 1: DEVE SER ATUALIZADO NORMALMENTE
UPDATE FUNC_PROJ
SET DtAlocIni = '10/10/2023', DataFimAlocInicial = '12/10/2023',
DataIniSegundaAloc = '22/10/2023', DtAlocFim = '29/10/2023'
WHERE IdFuncionario = 1

--TESTE 2: PROVOCANDO ERRO 1 -> DtAlocIni Deve ser maior ou igual
DtAdmissão
UPDATE FUNC_PROJ
SET DtAlocIni = '10/11/2014'
WHERE IdFuncionario = 1
SELECT * FROM FUNC_PROJ

--TESTE 3: PROVOCANDO ERRO 2 -> DtAlocFim deve ser no máximo igual
DtDemissão
UPDATE FUNCIONARIOS
SET DtDemissao = '12/12/2023'
WHERE IdFuncionario = 1
SELECT * FROM FUNCIONARIOS

UPDATE FUNC_PROJ
SET DtAlocFim = '30/12/2023'
WHERE IdFuncionario = 1
SELECT * FROM FUNC_PROJ

--TESTE 4: PROVOCANDO ERRO 3 -> DtAlocInicial deve ser a menor data
UPDATE FUNC_PROJ
SET DtAlocIni='12/12/2023', DtAlocFim = '02/12/2023'
WHERE IdFuncionario = 1
SELECT * FROM FUNC_PROJ

--TESTE 5: PROVOCANDO ERRO 4 -> DtFimAlocInicial deve ser maior que DtAlocInicial
e menor que
--DtInicioAlocSegunda e DtAlocFinal, DESDE QUE não seja NULO
--(2 intervalos de operação no projeto)
UPDATE FUNC_PROJ
SET DtAlocIni='01/12/2023', DataFimAlocInicial = '09/12/2023',
DataIniSegundaAloc = '08/12/2023', DtAlocFim = '11/12/2023'
WHERE IdFuncionario = 1
SELECT * FROM FUNC_PROJ

--TESTE 6: PROVOCANDO ERRO 5 -> DtInicioAlocSegunda deve ser maior que
DtAlocInicial e DtFimAlocInicial,
```

```
--e menor que a DtAlocFinal, DESDE QUE não seja nulo
--(2 intervalos de operação no projeto)
UPDATE FUNC_PROJ
    SET DtAlocIni='01/12/2023', DataFimAlocInicial = '04/12/2023',
    DataIniSegundaAloc = '07/12/2023', DtAlocFim = '06/12/2023'
    WHERE IdFuncionario = 1
SELECT * FROM FUNC_PROJ

--TESTE 7: PROVOCANDO ERRO 6 -> DtFimAloc deve ser a maior data SE NÃO NULO
UPDATE FUNC_PROJ
    SET DtAlocIni='01/12/2023', DataFimAlocInicial = '04/12/2023',
    DataIniSegundaAloc = '07/12/2023', DtAlocFim = '02/12/2023'
    WHERE IdFuncionario = 1

--TESTE 8: PROVOCANDO ERRO 7 -> Todas as datas deve ser maiores que a de inicio
do projeto, apenas a DtAlocInicial pode ser IGUAL
UPDATE FUNC_PROJ
    SET DtAlocIni='17/07/2023', DataFimAlocInicial = '04/12/2023',
    DataIniSegundaAloc = '07/12/2023', DtAlocFim = '12/12/2023'
    WHERE IdFuncionario = 1
SELECT * FROM FUNC_PROJ

--TESTE 9: PROVOCANDO ERRO 8 -> Todas as datas devem ser menores que a de fim do
projeto, apenas DtAlocFinal pode ser IGUAL
--Inserindo data final no projeto
UPDATE PROJETO
    SET DtFim = '10/12/2023'
    WHERE CodProj = 10

UPDATE FUNC_PROJ
    SET DtAlocIni='02/12/2023', DataFimAlocInicial = '04/12/2023',
    DataIniSegundaAloc = '07/12/2023', DtAlocFim = '12/12/2023'
    WHERE IdFuncionario = 1
SELECT * FROM FUNC_PROJ
```

9. Elabore uma rotina que imprima relatório em 3 colunas referente a tabela de Cargos. O lay-out poderá ser o que segue:

Relação de Cargos

Cod.cargo	Descrição	Cod.cargo	Descrição	Cod.cargo	Descrição
999	XXXXXXXX	999	XXXXXXXX	999	XXXXXXXX
999	XXXXXXXX	999	XXXXXXXX	999	XXXXXXXX
999	XXXXXXXX	999	XXXXXXXX	999	XXXXXXXX
999	XXXXXXXX	999	XXXXXXXX	999	XXXXXXXX
Total de Cargos:		9.999			

Não compreendi o que deveria ser exibido como relatório, se deveria aparecer 3 colunas iguais, 3 colunas com dados diferentes... Para sua resolução, criei uma nova tabela denominada “Relatorio” e passei para a mesma os valores de CodCargo e Descrição da tabela cargo, utilizando uma procedure. Como a tabela “Relatório” é temporária, ao final, seus dados são eliminados pois ela serve apenas para representação no select. O resultado foi:

```

1 CREATE TABLE RELATORIO
2 (
3     IdCargo INT,
4     DescCargo TEXT,
5     IdCargo2 INT,
6     DescCargo2 TEXT,
7     IdCargo3 INT,
8     DescCargo3 TEXT,
9
10    PRIMARY KEY (IdCargo),
11 );
12

```



```

CREATE PROCEDURE RELATORIO_IMPRIMIR
AS
BEGIN
    --Passando os dados de CARGO para RELATORIO

    INSERT INTO RELATORIO (IdCargo, DescCargo, IdCargo2, DescCargo2, IdCargo3, DescCargo3)
    SELECT
        C1.IdCargo,
        C1.Descr_Cargo,
        C2.IdCargo,
        C2.Descr_Cargo,
        C3.IdCargo,
        C3.Descr_Cargo
    FROM
        (SELECT *, ROW_NUMBER() OVER (ORDER BY IdCargo) AS linha FROM CARGO) C1
    LEFT JOIN
        (SELECT *, ROW_NUMBER() OVER (ORDER BY IdCargo) AS linha FROM CARGO) C2
        ON C1.linha = C2.linha - 1
    LEFT JOIN
        (SELECT *, ROW_NUMBER() OVER (ORDER BY IdCargo) AS linha FROM CARGO) C3
        ON C1.linha = C3.linha - 2;

    SELECT * FROM RELATORIO

    SELECT COUNT(*) AS TotalCargos
    FROM CARGO;

    DELETE RELATORIO

END;

EXEC RELATORIO_IMPRIMIR

```

112 %

Resultados

Mensagens

	IdCargo	DescCargo	IdCargo2	DescCargo2	IdCargo3	DescCargo3
1	111	Chefe	222	Analista de Redes	333	Programador
2	222	Analista de Redes	333	Programador	444	Consultor financeiro
3	333	Programador	444	Consultor financeiro	555	Gerente de Ativos
4	444	Consultor financeiro	555	Gerente de Ativos	NULL	NULL
5	555	Gerente de Ativos	NULL	NULL	NULL	NULL

	TotalCargos
1	5

CÓDIGO PROCEDURE RELATÓRIO

```
CREATE TABLE RELATORIO
(
    IdCargo INT,
    DescCargo TEXT,
    IdCargo2 INT,
    DescCargo2 TEXT,
    IdCargo3 INT,
    DescCargo3 TEXT,

    PRIMARY KEY (IdCargo),
);

CREATE PROCEDURE RELATORIO_IMPRIMIR

AS
BEGIN

    --Passando os dados de CARGO para RELATORIO

    INSERT INTO RELATORIO (IdCargo, DescCargo, IdCargo2, DescCargo2,
IdCargo3, DescCargo3)
    SELECT
        C1.IdCargo,
        C1.Descr_Cargo,
        C2.IdCargo,
        C2.Descr_Cargo,
        C3.IdCargo,
        C3.Descr_Cargo
    FROM
        (SELECT *, ROW_NUMBER() OVER (ORDER BY IdCargo) AS linha FROM
CARGO) C1
    LEFT JOIN
        (SELECT *, ROW_NUMBER() OVER (ORDER BY IdCargo) AS linha FROM
CARGO) C2
        ON C1.linha = C2.linha - 1
    LEFT JOIN
        (SELECT *, ROW_NUMBER() OVER (ORDER BY IdCargo) AS linha FROM
CARGO) C3
        ON C1.linha = C3.linha - 2;

    SELECT * FROM RELATORIO

    SELECT COUNT(*) AS TotalCargos
    FROM CARGO;

    DELETE RELATORIO

END;

EXEC RELATORIO_IMPRIMIR
```

Referências Bibliográficas:

MIKERAYMSFT. **Tipos de dados (Transact-SQL) - SQL Server**. Disponível em:
<<https://learn.microsoft.com/pt-br/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver16>>.

MARKINGMYNAME. CAST e CONVERT (Transact-SQL) - SQL Server.

Disponível em: <<https://learn.microsoft.com/pt-br/sql/t-sql/functions/cast-and-convert-transact-sql?view=sql-server-ver16>>. Acesso em: 20 out. 2023.

SHADAB. **Difference between Union And Join in SQL**. Disponível em:
<<https://atechdaily.com/posts/Difference-between-Join-And-Union-in-SQL?q=Best+Selling+Smartphone#gsc.tab=0&gsc.q=Best%20Selling%20Smartphone&gsc.page=1>>. Acesso em: 20 out. 2023.