

Pontifícia Universidade Católica de Minas Gerais - Belo Horizonte - MG – Brasil

BANCO DE DADOS II



SISTEMAS DE INFORMAÇÃO

EXERCICIOS SOBRE PROGRAMAÇÃO SGBD

Amanda Paula Campos

Guilherme Alves Sotero

Filipe Eduardo da Silva

Paulo Cezar Paim

Vinicius Soares Dias

Belo Horizonte

2017

2.1)crie um código que declara uma variável para receber um retorno de um SQL

```
DECLARE @Idpessoa AS INT, @Nome AS CHAR(50), @identidade varchar(50)
```

```
SET @identidade = 'M- 706.992'
```

```
SELECT @Nome = nome FROM pessoa where identidade > @identidade
```

```
print @identidade print @nome
```

2.2) crie um código que usa estrutura de controle if - else

```
DECLARE @Idpessoas int
```

```
set @Idpessoas = 2
```

```
IF @Idpessoas > 0
```

```
BEGIN
```

```
    SELECT * FROM pessoa WHERE idpessoa = @Idpessoas
```

```
END
```

```
ELSE
```

```
BEGIN
```

```
    SELECT * from pessoa
```

```
END
```

2.3) crie um código que usa estrutura de case

```
SELECT Nome, logradouro,
```

```
CASE profissao
```

```
    WHEN 'ESTUDANTE'
```

```
        then 'promissora'
```

```
    WHEN 'ARQUITETA'
```

```
        then 'interessante'
```

```
    WHEN 'PSICOLOGA'
```

```
        then 'indicada'
```

```
    else 'comum'
```

```
    end AS perspectiva
```

```
from pessoa
```

2.4) crie um código que faça o seguinte:

Declare uma variável que vai receber um nome de uma pessoa ok

Identifique a qte de vezes que essa pessoa pegou chaves

Se a qte for maior maior ou igual a 1

verificar se ele teve atendimento.

Se teve listar o corretor que o atendeu.

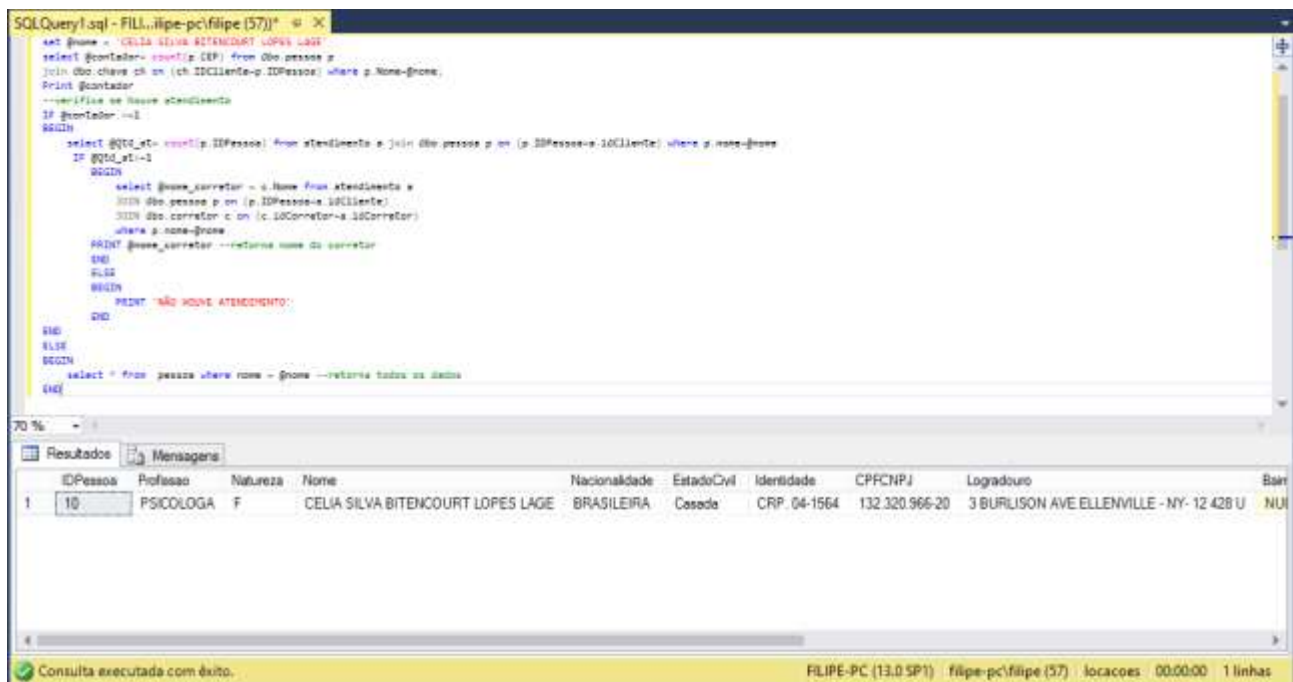
Se não listar “ não teve Atendimento.

Se a qte for igual a zeros, listar os seus dados (todos os atributos)

Entregar o código e o print de execução

```
DECLARE @nome char(50) , @nome2 char (50), @contador int, @Qtd_at int, @nome_corretor char(50)
```

```
set @nome = 'CELIA SILVA BITENCOURT LOPES LAGE'
select @contador= count(p.CEP) from dbo.pessoa p
join dbo.chave ch on (ch.IDCliente=p.IDPessoa) where p.Nome=@nome;
Print @contador
--verifica se houve atendimento
IF @contador >=1
BEGIN
    select @Qtd_at= count(p.IDPessoa) from atendimento a join dbo.pessoa p on
    (p.IDPessoa=a.idCliente) where p.nome=@nome
    IF @Qtd_at>=1
    BEGIN
        select @nome_corretor = c.Nome from atendimento a
        JOIN dbo.pessoa p on (p.IDPessoa=a.idCliente)
        JOIN dbo.corretor c on (c.idCorretor=a.idCorretor)
        where p.nome=@nome
        PRINT @nome_corretor --retorna nome do corretor
    END
    ELSE
    BEGIN
        PRINT 'NÃO HOUVE ATENDIMENTO'
    END
END
ELSE
BEGIN
    select * from pessoa where nome = @nome --retorna todos os dados
END
```



3.1) Criar um Stored Procedure para retornar a média dos” valores sugeridos de alugueis” para os imóveis por ano.

```
IF OBJECT_ID ( 'media_valor', 'P' ) IS NOT NULL
DROP PROCEDURE media_valor;
```

```

GO
CREATE PROCEDURE media_valor
AS

    select year(datacaptacao) as ano, avg (valorsugerido) as media from imovel
    group by year(datacaptacao)
GO
exec media_valor;

```

3.2) Criar um Stored Procedure para retornar a média dos valores sugeridos de alugueis para os imóveis por ano. O ano deve ser um parâmetro

```

IF OBJECT_ID ( 'media_valor_ano', 'P' ) IS NOT NULL
    DROP PROCEDURE media_valor_ano;
GO
CREATE PROCEDURE media_valor_ano @ano int
AS

    select year(datacaptacao) as ano, avg (valorsugerido) as media from imovel where
    year(datacaptacao)= @ano
    group by year(datacaptacao)
GO
exec media_valor_ano 2010;

```

3.3) Crie stored procedures que lista os dados da pessoa, a data dos seus atendimentos e o corretor que oo atendeu. Deve ser informado o nome do proprietário.

```

IF OBJECT_ID ( 'atendimentosporpessoa', 'P' ) IS NOT NULL
    DROP PROCEDURE atendimentosporpessoa;
GO
CREATE PROCEDURE atendimentosporpessoa
    @nome varchar(40)

AS

    select idpessoa, p. nome as pessoa, data as 'dataatendimento', c.nome as corretor from
    atendimento a, pessoa p, corretor c
    where a.idcliente=idpessoa and a.idcorretor=c.idcorretor and p.nome=@nome
GO
exec atendimentosporpessoa 'ANITA GARIBALDE';

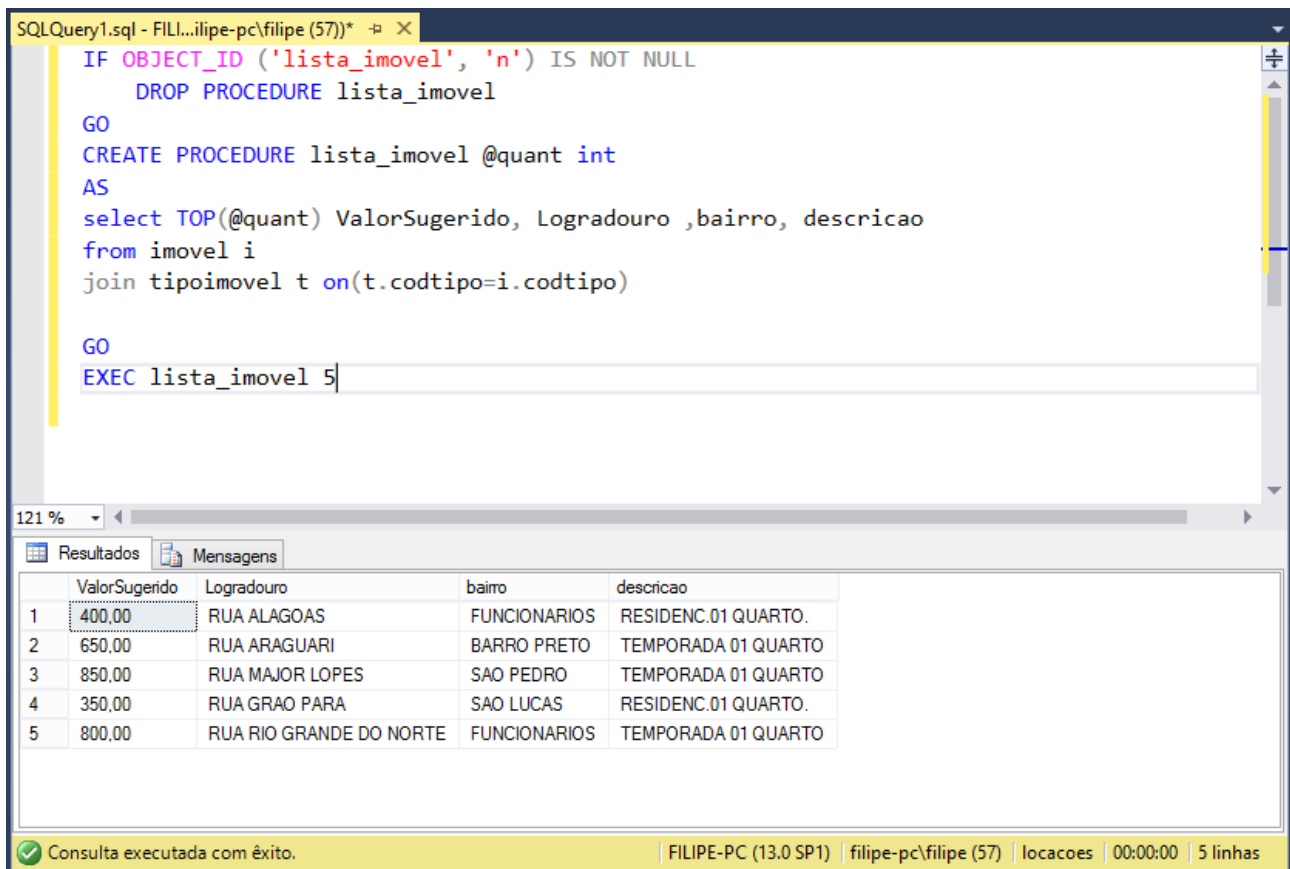
```

3.4) crie uma stored procedure que liste para cada imóvel a: “descrição” da tabela tipo de imóvel e da tabela imóvel o bairro, o logradouro e o valor sugerido para o aluguel. Só listar os ‘n’ imóveis que têm o maior valor sugerido. O valor n deve ser passado como parâmetro. Usar função top()

Entregar o código e o print de execução

```
IF OBJECT_ID ('lista_imovel', 'n') IS NOT NULL
    DROP PROCEDURE lista_imovel
GO
CREATE PROCEDURE lista_imovel @quant int
AS
select TOP(@quant) ValorSugerido, Logradouro ,bairro, descricao
from imovel i
join tipoimovel t on(t.codtipo=i.codtipo)

GO
EXEC lista_imovel 5
```



The screenshot shows a SQL Server window titled "SQLQuery1.sql - FILI...ilipe-pc\filipe (57)*". The query editor contains the following SQL code:

```
IF OBJECT_ID ('lista_imovel', 'n') IS NOT NULL
    DROP PROCEDURE lista_imovel
GO
CREATE PROCEDURE lista_imovel @quant int
AS
select TOP(@quant) ValorSugerido, Logradouro ,bairro, descricao
from imovel i
join tipoimovel t on(t.codtipo=i.codtipo)

GO
EXEC lista_imovel 5
```

Below the query editor, the "Resultados" (Results) tab is active, displaying a table with 5 rows and 4 columns: ValorSugerido, Logradouro, bairro, and descricao. The data is as follows:

	ValorSugerido	Logradouro	bairro	descricao
1	400,00	RUA ALAGOAS	FUNCIONARIOS	RESIDENC.01 QUARTO.
2	650,00	RUA ARAGUARI	BARRO PRETO	TEMPORADA 01 QUARTO
3	850,00	RUA MAJOR LOPES	SAO PEDRO	TEMPORADA 01 QUARTO
4	350,00	RUA GRAO PARA	SAO LUCAS	RESIDENC.01 QUARTO.
5	800,00	RUA RIO GRANDE DO NORTE	FUNCIONARIOS	TEMPORADA 01 QUARTO

At the bottom of the window, a status bar indicates: "Consulta executada com êxito." (Query executed successfully). Other information in the status bar includes: "FILIPE-PC (13.0 SP1)", "filipe-pc\filipe (57)", "locacoes", "00:00:00", and "5 linhas".

3.5) implemente o exercício 2.4 como stored procedure

```
IF OBJECT_ID ( 'retorno_dados', 'P' ) IS NOT NULL
    DROP PROCEDURE retorno_dados;
GO
CREATE PROCEDURE retorno_dados @nome_pesquisa char(90)
AS
    DECLARE @nome char(50) , @contador int, @Qtd_at int, @nome_corretor char(50)
    select @contador= count(p.CEP) from dbo.pessoa p
    join dbo.chave ch on (ch.IDCliente=p.IDPessoa) where p.Nome=@nome_pesquisa;
    Print @contador
    --verifica se houve atendimento
    IF @contador >=1
    BEGIN
        select @Qtd_at= count(p.IDPessoa) from atendimento a join dbo.pessoa p on
        (p.IDPessoa=a.idCliente) where p.nome=@nome_pesquisa
        IF @Qtd_at>=1
        BEGIN
            select @nome_corretor = c.Nome from atendimento a
            JOIN dbo.pessoa p on (p.IDPessoa=a.idCliente)
            JOIN dbo.corretor c on (c.idCorretor=a.idCorretor)
            where p.nome=@nome_pesquisa
            PRINT 'NOME DD'+@nome_pesquisa
        PRINT @nome_corretor --retorna nome do corretor
        END
        ELSE
        BEGIN
            PRINT 'NÃO HOUVE ATENDIMENTO'
        END
    END
    ELSE
    BEGIN
        select * from pessoa where nome = @nome_pesquisa --retorna todos os dados
    END

exec retorno_dados 'CELIA SILVA BITENCOURT LOPES LAGE'
```

```

SQLQuery3.sql - FILIPE-PC\Filipe (57)
CREATE PROCEDURE retorno_dados @nome_pesquisa char(90)
AS
DECLARE @nome char(50), @contador int, @Qtd_at int, @nome_corretor char(50)
-- set @nome = 'CELIA SILVA BITENCOURT LOPES LAGE'
select @contador = count(p.CEP) from dbo.pessoa p
join dbo.chave ch on (ch.IDCliente=p.IDPessoa) where p.Nome=@nome_pesquisa
Print @contador
--verifica se houve atendimento
IF @contador >=1
BEGIN
select @Qtd_at = count(p.IDPessoa) from atendimento a join dbo.pessoa p on (p.IDPessoa=a.idCliente) where p.nome=@nome_pesquisa
IF @Qtd_at >=1
BEGIN
select @nome_corretor = c.Nome from atendimento a
JOIN dbo.pessoa p on (p.IDPessoa=a.idCliente)
JOIN dbo.corretor c on (c.idCorretor=a.idCorretor)
where p.nome=@nome_pesquisa
PRINT 'NOME DO' + @nome_pesquisa
PRINT @nome_corretor --retorna nome do corretor
END
ELSE
BEGIN
PRINT 'NÃO HOUVE ATENDIMENTO'
END
END
ELSE
BEGIN
select * from pessoa where nome = @nome_pesquisa --retorna todos os dados
END
END

exec retorno_dados 'CELIA SILVA BITENCOURT LOPES LAGE'

```

IDPessoa	Profissao	Natureza	Nome	Nacionalidade	EstadoCivil	Identidade	CPCNPJ	Logradouro	Bairro
1	10	PSICOLOGA	F	CELIA SILVA BITENCOURT LOPES LAGE	BRASILEIRA	Casada	CRP: 04-1564	132 320 966-20	3 BURLISON AVE ELLENVILLE - NY- 12 428 U

Consulta executada com êxito. FILIPE-PC (13.0 SP1) | filipe-pc/filipe (57) | locais: 00:00:00 | 1 linhas

4.1) crie uma função que retorne uma tabela que tenha o nome da pessoa, a sua profissão e o corretor que o atendeu

```

DROP FUNCTION dbo.fpessoa
GO
CREATE FUNCTION dbo.fpessoa (@nome varchar(20))
RETURNS TABLE
AS
RETURN
(
    Select p.nome as pessoa, profissao, c.nome from atendimento a, pessoa p, corretor c
    where p.nome= @nome and p.idpessoa=a.idcliente and a.idcorretor=c.idcorretor )
GO

select * from fpessoa ('ANITA GARIBALDE')

```

4.2) Crie função que retorne a qte de imóveis de um bairro

```

DROP FUNCTION dbo.fimovelbairro
GO
CREATE FUNCTION dbo. fimovelbairro (@bairro varchar(20))
RETURNS int
AS
begin

```

```

declare @qte int;
select @qte= COUNT(*) from imovel
where bairro = @bairro
return @qte
end

```

GO

```
select dbo. fimovelbairro ('CENTRO')
```

4.3) crie uma função que retorne uma tabela com os campos do nome da pessoa, o responsável da entrega da chave e dados do imóvel. Só listar para os imóveis que tiveram entrega de chaves. Entregar o código e o print de execução .

GO

```
CREATE FUNCTION dbo.entrega ()
```

RETURNS TABLE

AS

```
RETURN(
```

```
    select p.Nome, ch.Responsavel,i.Logradouro, Numero,i.Cidade,i.Estado,i.Cep from
    dbo.pessoa p
```

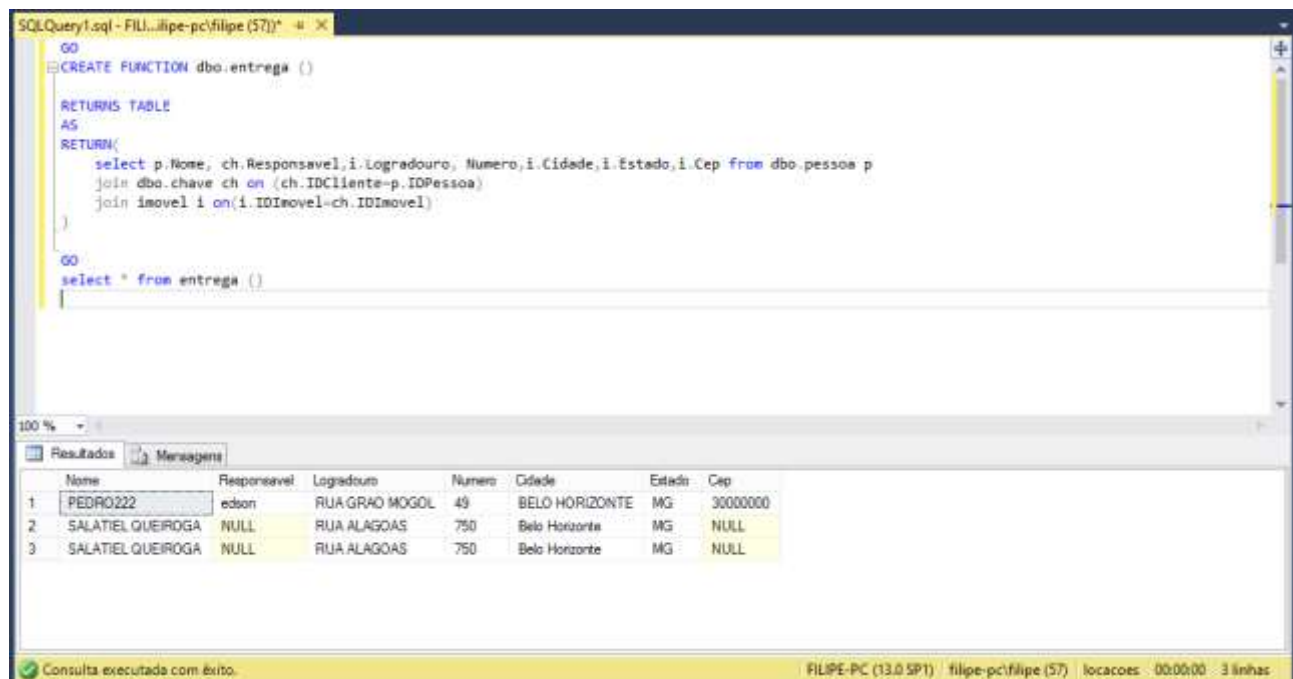
```
    join dbo.chave ch on (ch.IDCliente=p.IDPessoa)
```

```
    join imovel i on(i.IDImovel=ch.IDImovel)
```

```
)
```

GO

```
select * from entrega ()
```



The screenshot shows a SQL Server query window with the following code:

```

GO
CREATE FUNCTION dbo.entrega ()
RETURNS TABLE
AS
RETURN(
    select p.Nome, ch.Responsavel,i.Logradouro, Numero,i.Cidade,i.Estado,i.Cep from dbo.pessoa p
    join dbo.chave ch on (ch.IDCliente=p.IDPessoa)
    join imovel i on(i.IDImovel=ch.IDImovel)
)
GO
select * from entrega ()

```

Below the query window, the 'Results' pane displays the output of the query as a table with 7 columns: Nome, Responsavel, Logradouro, Numero, Cidade, Estado, and Cep. The table contains 3 rows of data.

	Nome	Responsavel	Logradouro	Numero	Cidade	Estado	Cep
1	PEDRO222	edson	RUA GRAO MOGOL	49	BELO HORIZONTE	MG	30000000
2	SALATIEL QUEIROGA	NULL	RUA ALAGOAS	750	Belo Horizonte	MG	NULL
3	SALATIEL QUEIROGA	NULL	RUA ALAGOAS	750	Belo Horizonte	MG	NULL

At the bottom of the window, a status bar indicates: 'Consulta executada com êxito.' (Query executed successfully).

4.4) Crie uma função que retorne os corretores e a qte de atendimento que fizeram


```

DROP FUNCTION dbo.vendaCorretor
GO
CREATE FUNCTION dbo.vendaCorretor ()
RETURNS TABLE
AS
RETURN(select c.Nome , count(a.IDAtendimento) as Quantidade from corretor c join atendimento
a on (a.idCorretor=c.idCorretor) group by nome)
GO

select * from vendaCorretor()

```



5) exercícios sobre trigger

5.1) Crie uma trigger que sempre que uma ação de inclusão, alteração ou exclusão for executada na tabela Infracao, uma mensagem seja exibida. (usar RAISERROR)

```

DROP trigger notificacao
go
CREATE TRIGGER notificacao

```

```

ON pessoa
AFTER INSERT, UPDATE, delete
AS RAISERROR ('pessoa alterada', 16, 10);
GO
update pessoa set nome = 'p1' where nome = 'p7'

```

5.2) Crie uma trigger que sempre liste o nome e o endereço do proprietário antes e depois de uma alteração

```
drop trigger notificacao2
go
CREATE TRIGGER notificacao2
ON pessoa
AFTER UPDATE
AS
if (select COUNT(*) from deleted) > 0
select nome, profissao, 'antigo' from deleted
select nome, profissao, 'novo' from inserted
```

GO

```
update pessoa set nome = 'MARIA ' where nome = 'HELENA BARBOSA GOMIDE'
```

5.3) Crie uma trigger que sempre que uma ação de alteração for executada na tabela imóvel, a tabela auditoria_imovel deverá ser atualizada. A tabela auditoria deve ser criada. Ela deve ter os atributos da tabela imóvel, a data de alteração e quem fez a alteração.

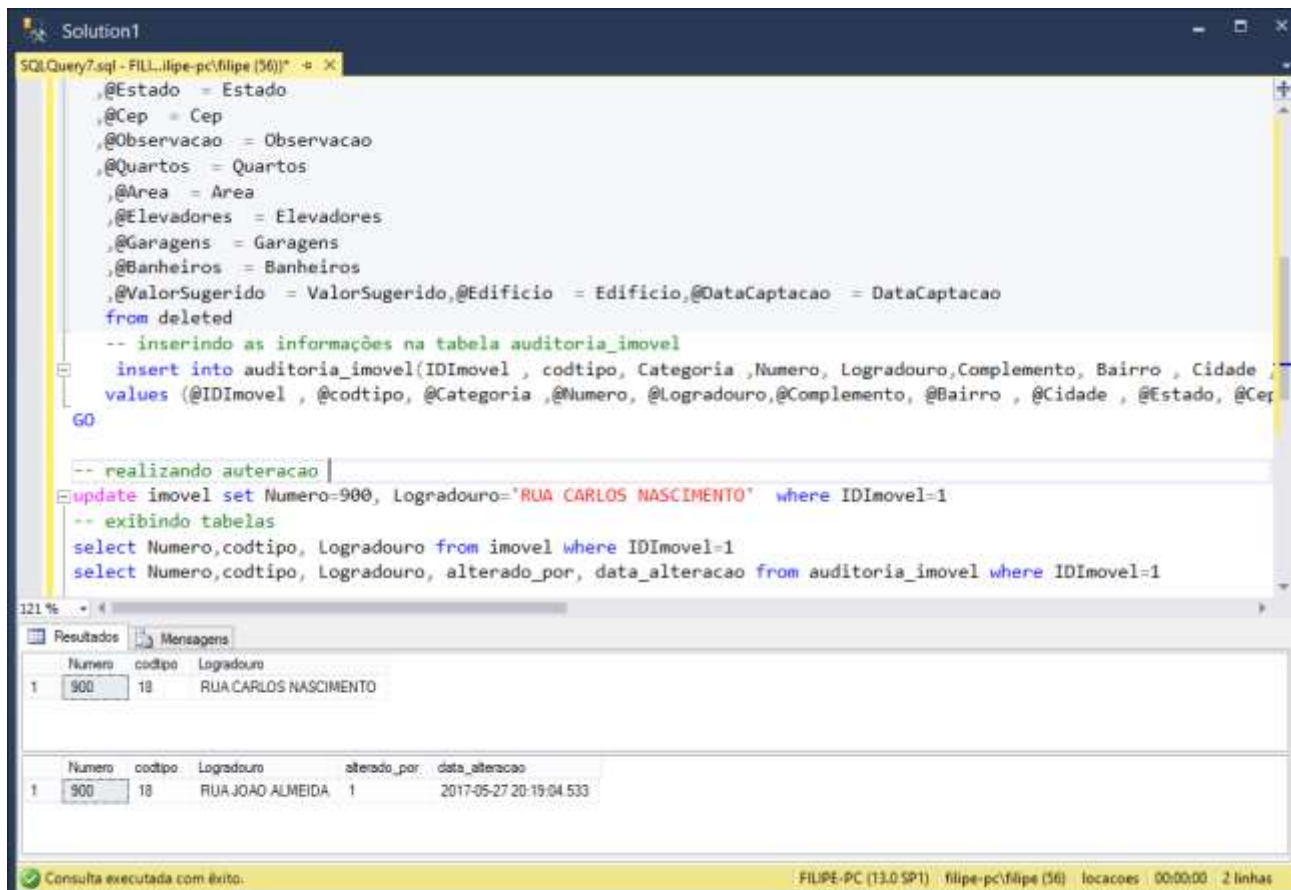
```
drop trigger t_auditoria_imovel
go

CREATE TRIGGER t_auditoria_imovel
ON imovel
AFTER UPDATE
AS
DECLARE @IDImovel int, @codtipo int, @Categoria nvarchar(1), @Numero nvarchar(10),
        @Logradouro nvarchar(35), @Complemento nvarchar(10), @Bairro nvarchar(20),
        @Cidade nvarchar(20), @Estado nvarchar(2), @Cep nvarchar(9),
        @Observacao nvarchar, @Quartos tinyint, @Area smallint, @Elevadores tinyint,
        @Garagens tinyint,
        @Banheiros tinyint, @ValorSugerido money, @Edificio nvarchar(30), @DataCaptacao
        datetime, @dataAlteracao datetime, @alterado_por nvarchar(40)

set @dataAlteracao = GETDATE()
set @alterado_por = 1 -- RECEBE ID DO USUÁRIO QUE ALTEROU A INFORMAÇÃO

select @IDImovel = IDImovel, @codtipo = codtipo, @Categoria = Categoria, @Numero =
Numero, @Logradouro = Logradouro, @Complemento = Complemento
        , @Bairro = Bairro, @Cidade = Cidade, @Estado = Estado, @Cep = Cep, @Observacao
= Observacao, @Quartos = Quartos, @Area = Area, @Elevadores = Elevadores,
@Garagens = Garagens, @Banheiros = Banheiros, @ValorSugerido = ValorSugerido, @Edificio =
Edificio, @DataCaptacao = DataCaptacao
        from deleted
        -- inserindo as informações na tabela auditoria_imovel
        insert into auditoria_imovel(IDImovel, codtipo, Categoria, Numero,
Logradouro, Complemento, Bairro, Cidade, Estado, Cep, Observacao, Quartos, Area, Elevadores,
Garagens, Banheiros, ValorSugerido, Edificio, DataCaptacao, data_alteracao, alterado_por)
        values (@IDImovel, @codtipo, @Categoria, @Numero, @Logradouro, @Complemento, @Bairro,
        @Cidade, @Estado, @Cep, @Observacao, @Quartos, @Area, @Elevadores, @Garagens, @Banheiros,
        @ValorSugerido, @Edificio, @DataCaptacao, @dataAlteracao, @alterado_por)
GO
```

```
-- realizando auteracao
update imovel set Numero=900, Logradouro='RUA CARLOS NASCIMENTO' where IDImovel=1
-- exibindo tabelas
select Numero,codtipo, Logradouro from imovel where IDImovel=1
select Numero,codtipo, Logradouro, alterado_por, data_alteracao from auditoria_imovel where IDImovel=1
```



5.4) Crie uma trigger que antes que uma pessoa seja excluída mostre uma mensagem dizendo que aquela pessoa está sendo excluída. Mostre o nome, a profissão, a data e quem está excluindo.

```
drop trigger deleta_pessoa
go
CREATE TRIGGER deleta_pessoa
ON pessoa
AFTER delete
AS
if (select COUNT(*) from deleted) > 0
    DECLARE @nome char(50), @profissao char(50), @data datetime
    set @data = (select DATEADD(day, - DATEPART(day, getdate()) + 1, getdate()))
    select @nome = nome, @profissao = profissao from deleted
    print 'ESSA PESSOA ESTÁ SENDO EXCLUÍDA'
    PRINT 'NOME: ' + @nome
```

```

PRINT 'Profissao: ' + @profissao
PRINT 'DATA: '
print @data

```

```
delete pessoa where Nome='MARIA LUCIA LINDGREN ALVES';
```

The screenshot shows a SQL Server Enterprise Manager window with two tabs: 'SQLQuery5.sql - FILI...ilipe-pc\filipe (57)*' and 'SQLQuery2.sql - FILI...ilipe-pc\filipe (59)*'. The active tab shows a SQL script for creating a trigger named 'deleta_pessoa' on the 'pessoa' table. The trigger is an AFTER DELETE trigger that checks if any rows were deleted. If so, it declares variables for name, profession, and date, then sets the date to one day after the delete operation. It then selects the deleted row's data and prints it. The script concludes with a DELETE statement for 'MARIA LUCIA LINDGREN ALVES'. Below the script, the 'Mensagens' (Messages) pane shows the output of the trigger: 'ESSA PESSOA ESTÁ SENDO EXCLUÍDA', 'NOME: MARIA LUCIA LINDGREN ALVES', 'Profissao: CONTADORA', and 'DATA: Mai 1 2017 3:57PM'. It also indicates that 2 lines were affected. The status bar at the bottom shows 'Consulta executada com êxito.' (Query executed successfully).

```

drop trigger deleta_pessoa
go
CREATE TRIGGER deleta_pessoa
ON pessoa
AFTER delete
AS
if (select COUNT(*) from deleted) > 0
DECLARE @nome char(50), @profissao char(50), @data datetime
set @data = (select DATEADD(day, - DATEPART(day, getdate()) + 1, getdate()))
select @nome = nome, @profissao = profissao from deleted
print 'ESSA PESSOA ESTÁ SENDO EXCLUÍDA'
PRINT 'NOME: ' + @nome
PRINT 'Profissao: ' + @profissao
PRINT 'DATA: '
print @data

delete pessoa where Nome='MARIA LUCIA LINDGREN ALVES';

```

Mensagens

```

ESSA PESSOA ESTÁ SENDO EXCLUÍDA
NOME: MARIA LUCIA LINDGREN ALVES
Profissao: CONTADORA
DATA:
Mai 1 2017 3:57PM

(2 linha(s) afetadas)

```

Consulta executada com êxito. | FILIPE-PC (13.0 SP1) | filipe-pc\filipe (57) | locacoes | 00:00:00 | 0 linhas

5.5) Crie uma trigger que não deixe o valor sugerido para o imóvel ser alterado para um valor maior que 10.000,00. Se isso acontecer, retornar uma mensagem “valor inválido” . mostre também o valor atual e o valor pretendido.

```

drop trigger altera_valor
go
CREATE TRIGGER altera_valor
ON imovel
FOR update
AS
DECLARE @VALOR money, @VALOR_INSERIDO money
SELECT @VALOR_INSERIDO = valorSugerido FROM inserted
select @VALOR = valorSugerido FROM deleted

IF (@VALOR_INSERIDO > 10000.00)
BEGIN
    PRINT 'VALOR INSERIDO: '
    PRINT @VALOR_INSERIDO
    PRINT 'VALOR ATUAL: '
    PRINT @VALOR

    RAISERROR (VALOR INVÁLIDO ', 1, 1)
    ROLLBACK
    TRAN
END

update imovel set ValorSugerido = 10001.00 where IDImovel=1

```

SQLQuery7.sql - FILIPE-PC.locacoes (filipe-pc\filipe (56))*

```
drop trigger altera_valor
go
CREATE TRIGGER altera_valor
ON imovel
FOR update
AS
    DECLARE @VALOR money, @VALOR_INSERIDO money
    SELECT @VALOR_INSERIDO = valorSugerido FROM inserted
    select @VALOR = valorSugerido FROM deleted

    IF(@VALOR_INSERIDO>10000.00)
    BEGIN
        PRINT 'VALOR ATUAL:'
        PRINT 'VALOR INSERIDO: '
        PRINT @VALOR_INSERIDO
        PRINT 'VALOR ATUAL:'
        PRINT @VALOR
        RAISERROR ('VALOR INVÁLIDO',1,1)
        ROLLBACK
        TRAN
    END

    update imovel set ValorSugerido =90000.00 where IDImovel=1
```

100 %

Mensagens

```
(1 linha(s) afetadas)
VALOR ATUAL:
VALOR INSERIDO:
90000.00
VALOR ATUAL:
120.00
VALOR INVÁLIDO
Mensagem 50000, Nível 1, Estado 1
Mensagem 3609, Nível 16, Estado 1, Linha 23
A transação foi encerrada no gatilho. O lote foi anulado.
```

100 %