```
/* Criar o diretório C:\FBD

CREATE DATABASE loja
ON
         PRIMARY
         (
         NAME = 'loja',
         FILENAME = ‘C:\FBD\loja.mdf',
         SIZE = 5120KB,
         FILEGROWTH = 1024KB
         ),

         FILEGROUP loja_fg01
         (
         NAME = 'loja_001',
         FILENAME = ‘C:\FBD\loja_001.ndf',
         SIZE = 1024KB,
         FILEGROWTH = 30%
         ),
         (
         NAME ='loja_002',
         FILENAME = ‘C:\FBD\loja_002.ndf',
         SIZE = 1024KB,
         MAXSIZE = 3072KB,
         FILEGROWTH = 15%
         ),

         FILEGROUP loja_fg02
         (
         NAME = 'loja_003',
         FILENAME = ‘C:\FBD\loja_003.ndf',
         SIZE = 2048KB,
         MAXSIZE = 5120KB,
         FILEGROWTH = 1024KB
         )

         LOG ON
         (
         NAME = 'loja_log',
         FILENAME = ‘C:\FBD\loja_log.ldf',
         SIZE = 1024KB,
         FILEGROWTH = 10%
         )
-------------------------------------------------------------------
-------
USE loja

CREATE TABLE filiais
         (
         cod smallint NOT NULL,
         nome nvarchar(50) NOT NULL,
         cid nvarchar(50) NOT NULL,

         CONSTRAINT filiais_PK PRIMARY KEY (cod),
```

```
        )  ON loja_fg01

CREATE TABLE fornecedores
        (
        cod smallint NOT NULL,
        nome nvarchar(50) NOT NULL,
        cid nvarchar(50) NOT NULL,

        CONSTRAINT fornecedores_PK PRIMARY KEY (cod),
        )  ON loja_fg01

CREATE TABLE estoque
        (
        cod int,
        ref nvarchar(50) NOT NULL,
        prcom decimal(6, 2) NOT NULL,
        prven decimal(6, 2) NOT NULL,
        qtde smallint NOT NULL,
        codfor smallint NOT NULL,

        CONSTRAINT estoque_PK PRIMARY KEY NONCLUSTERED (cod),
        CONSTRAINT estoque_CK_qtde CHECK (qtde >= 0),
        CONSTRAINT estoque_CK_prven CHECK (prven > prcom),
        CONSTRAINT estoque_FK_codfor FOREIGN KEY (codfor)
REFERENCES fornecedores (cod) ON UPDATE NO ACTION ON DELETE NO
ACTION
        )  ON loja_fg02

CREATE TABLE vendedores
        (
        matr smallint,
        nome nvarchar(50) NOT NULL,
        salario decimal(6, 2) NOT NULL DEFAULT 900.00,
        codfil smallint NOT NULL,

        CONSTRAINT vendedores_PK PRIMARY KEY NONCLUSTERED (matr),
        CONSTRAINT vendedores_CK_salario CHECK (salario >= 900.00),
        CONSTRAINT vendedores_FK_codfil FOREIGN KEY (codfil)
REFERENCES filiais (cod) ON UPDATE cascade ON DELETE NO ACTION
        )  ON loja_fg02

CREATE CLUSTERED INDEX vendedores_IDX_sal
        ON vendedores (salario)
        WITH (fillfactor=100, pad_index=on)

CREATE TABLE historico
        (
        matrvend smallint NOT NULL,
        coditem int NOT NULL,
        qtde smallint NOT NULL,
        prven decimal(6,2) NOT NULL,
        dthoraven datetime DEFAULT GETDATE()-3,

        CONSTRAINT historico_FK_vendedor FOREIGN KEY (matrvend)
```

```
REFERENCES vendedores (matr) ON UPDATE NO ACTION ON DELETE NO
ACTION,
        CONSTRAINT historico_FK_item FOREIGN KEY (coditem)
REFERENCES estoque (cod) ON UPDATE  NO ACTION ON DELETE  NO ACTION,
        CONSTRAINT historico_CK_qtde CHECK (qtde >= 1)
        )  ON loja_fg02
-------------------------------------------------------------------
-------

CREATE VIEW view_vendedores_maiores_salarios
AS
        SELECT nome
        FROM vendedores
        GROUP BY nome
        HAVING AVG(sal) > (SELECT AVG(sal) FROM vendedores)

CREATE VIEW view_vendedores_nomes_maiores_sal (nome)
AS
        SELECT *
        FROM view_vendedores_maiores_salarios

CREATE VIEW view_vendedores_historico
AS
        SELECT
                v.nome as 'Vendedores', e.ref as 'Itens', h.qtde as
'Quantidade'
        FROM
                vendedores v, historico h, estoque e
        WHERE   v.matr = h.matrvend and
                h.coditem = e.cod

-------------------------------------------------------------------
-------

CREATE TRIGGER historico_TR_preco
       ON historico
       FOR INSERT, UPDATE
       AS

       IF (SELECT prven FROM inserted)
       NOT IN
       (SELECT e.prven FROM estoque e INNER JOIN inserted i ON
i.coditem = e.cod)

       BEGIN
               RAISERROR ('Preço de venda diferente do estipulado na
tabela ESTOQUE', 10, 6)
               ROLLBACK TRANSACTION
       END


CREATE TRIGGER historico_TR_qtde
       ON  historico
       AFTER INSERT, UPDATE, DELETE
```

```
       AS
       BEGIN
              SET NOCOUNT ON;
              UPDATE estoque SET qtde = e.qtde - i.qtde
              FROM estoque e INNER JOIN inserted i ON e.cod =
i.coditem

              UPDATE estoque SET qtde = e.qtde - d.qtde
              FROM estoque e INNER JOIN deleted d ON e.cod =
d.coditem
       END

CREATE TRIGGER estoque_TR_qtdelimite
        ON  estoque
        AFTER INSERT, UPDATE
        AS
                IF ( (SELECT qtde FROM estoque WHERE cod = (SELECT
cod FROM inserted)) < (100) )
        BEGIN
                print('Atenção! Quantidade de estoque baixa.')
        END

=====================================================================
====================
-- Retorna mÈdia de salarios da filial
-- com cod igual @codfil_input

create function f1 (@codfil_input smallint)
returns dec(10,2)
as
begin
declare @media dec(10,2)
select @media=avg(salario) from vendedores v
where v.codfil=@codfil_input
return (@media)
end
=====================================================================
========================
-- Parametro output -- Stored Procedure

create procedure cal_med_sal_fil2
@codfil smallint,
@media dec(10,3) output
as
select avg(salario) from vendedores v
where v.codfil=@codfil

create procedure chama_outra_proc
as
declare @media dec(10,3)
exec cal_med_sal_fil2 1, @media output
print @media

create procedure proc_lojas as
```

```
DECLARE cursor_lojas CURSOR SCROLL for
Select matr, v.nome, count(*)
from vendedores v left outer join historico h
on matr=matrvend
Group by matr, v.nome
order by 3 asc
DECLARE @matricula smallint, @qtde_vendas int,
@nome_vend char(20)
OPEN cursor_lojas
FETCH last FROM cursor_lojas
INTO @matricula, @nome_vend, @qtde_vendas
WHILE (@@fetch_status = 0)
BEGIN
PRINT 'Matricula:' + cast(@matricula as char(5))
 + ' - '+ 'Nome Vendedor: '+ @nome_vend + 'Quantidade de Vendas: ' +
 cast(@qtde_vendas as char(20))
FETCH prior FROM cursor_lojas
INTO @matricula, @nome_vend, @qtde_vendas
END
DEALLOCATE cursor_lojas
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
-- Gatilho para verificar se o novo salário não é maior que duas
vezes a média salarial da filial

create trigger [dbo].[sal_Maior_2_med] on [dbo].[vendedores] for
insert,update
 as
 if update(salario)
  begin
   declare @filial smallint
   declare @novo_salario dec(6,2)
   declare @media_salario_filial dec (9,2)
   declare @soma_salario_filial dec (9,2)
   declare @matr_vend smallint, @num_vend int
   declare @str_invalido char(46)
   DECLARE cursor_lojas CURSOR SCROLL for
   select matr, codfil, salario from inserted
   OPEN cursor_lojas
   FETCH first FROM cursor_lojas
    INTO @matr_vend, @filial, @novo_salario
   WHILE (@@fetch_status = 0)
   BEGIN
     select @soma_salario_filial=sum(v.salario), @num_vend=count(*)
      from vendedores v
      where v.codfil=@filial and v.matr<>@matr_vend
     set @media_salario_filial=@soma_salario_filial/@num_vend
     set @str_invalido='Salario maior que a media salarial da filial
'+cast(@filial as varchar(5))
     if (@novo_salario) > 2*(@media_salario_filial)
      begin
       raiserror(@str_invalido,16,1)
            rollback transaction
      end
```

```
          FETCH next FROM cursor_lojas
      INTO @matr_vend, @filial, @novo_salario
   end
   DEALLOCATE cursor_lojas
end
++++++++++++++++++++++++++++++++++++++++++++++++++++++++

-- Função que retorna uma tabela

create function f_item_vend_vendedor
(@coditem_ent smallint,@matr_ent smallint)
returns @tab_result table
(Nome_Item varchar(40),
Nome_Vend varchar(40),
qtde_vendida int)
as
begin
-- passar parametro nulo tem que ser entre ''
--if @coditem_ent=''
--  set @coditem_ent=1;
insert into @tab_result
select ref,nome,sum(h.qtde)
from estoque left outer join historico h
inner join vendedores on matr=matrvend on coditem=cod
where cod=@coditem_ent and matrvend=@matr_ent
group by cod,ref,matr,nome
return
end
++++++++++++++++++++++++++++++++++++++++++++++++++++++++

create procedure calc_comissao as
DECLARE cursor_lojas CURSOR SCROLL for
Select matr, v.nome, count(*)
from vendedores v left outer join historico h
on matr=matrvend
Group by matr, v.nome
order by 3 asc
DECLARE @matricula smallint, @qtde_vendas int,
@nome_vend char(20), @valor_comissao dec (10,2),
 @sal_vend dec(8,2)
OPEN cursor_lojas
FETCH last FROM cursor_lojas
INTO @matricula, @nome_vend, @qtde_vendas
WHILE (@@fetch_status = 0)
BEGIN
set @valor_comissao=@qtde_vendas*100
set @sal_vend=(select salario from vendedores where matr=@matricula)
if @valor_comissao>@sal_vend
 set @valor_comissao=@sal_vend;
PRINT 'Matricula:' + cast(@matricula as char(5))
 + ' - '+ 'Nome Vendedor: '+ @nome_vend + 'Quantidade de Vendas: ' +
 cast(@qtde_vendas as char(20)) +' '+ 'Valor Comissao: ' +
cast(@valor_comissao as char(20))
FETCH prior FROM cursor_lojas
```

```
INTO @matricula, @nome_vend, @qtde_vendas
END
DEALLOCATE cursor_lojas
```