

# Atividade

## *Trigger e stored procedures (automatização)*

Nesse conteúdo, você vai acompanhar como aplicar estruturas de automatização em seu banco de dados, aplicando *triggers* (gatilhos) e *stored procedures* (procedimentos armazenados).

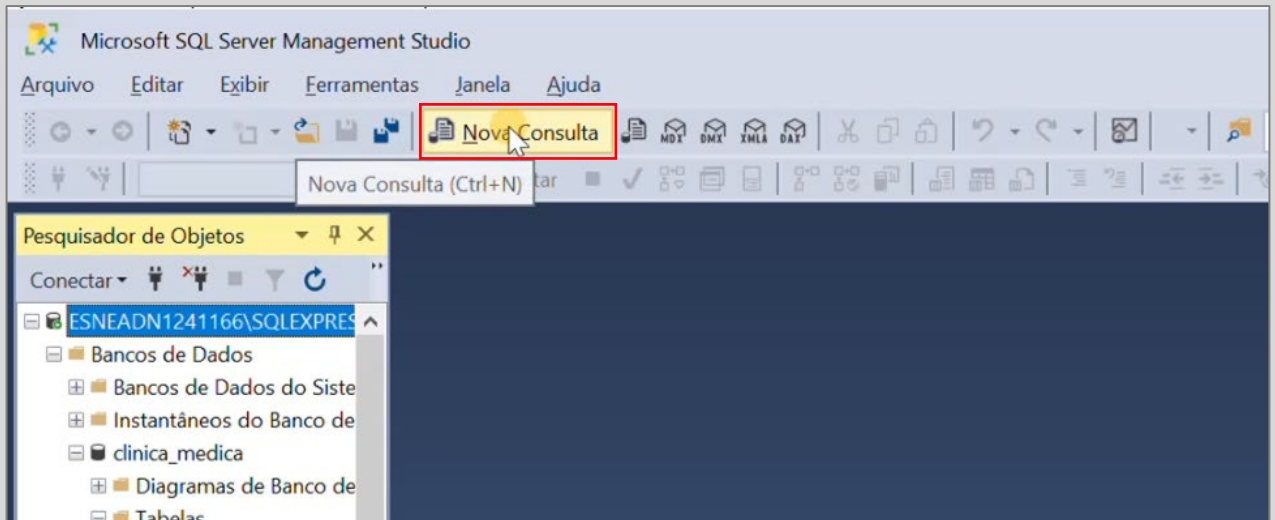
Para programar adequadamente esses procedimentos, devemos entender as demandas do projeto. Em nosso exemplo da clínica médica, as demandas são:

- a cada inclusão de pedido de exame, um gatilho (trigger) calcula o valor a ser pago e registra na própria tabela, seguindo o critério:
  - Plano de saúde Especial = 100% de desconto;
  - Plano de saúde Padrão = 30% de desconto;
  - Plano de saúde Básico = 10% de desconto;
- criação de procedimentos armazenados (stored procedures) para consulta de agenda de médicos, exames solicitados, histórico de pagamentos e resumo de pagamentos.

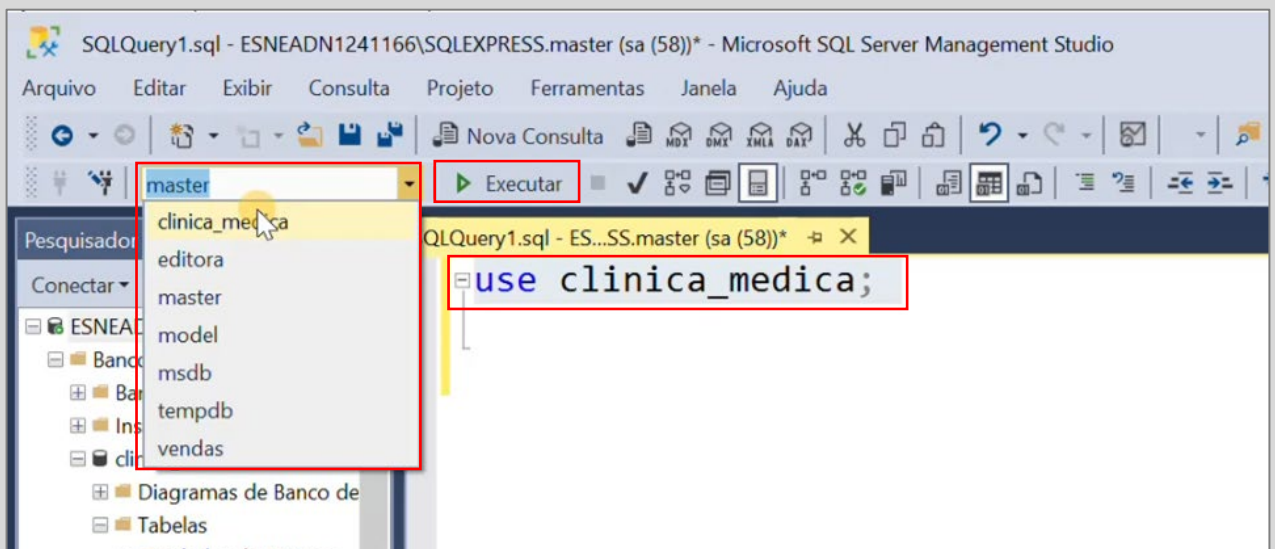
Para essa atividade, vamos usar o SSMS (SQL Server Management Studio da Microsoft).

## Trigger (gatilho)

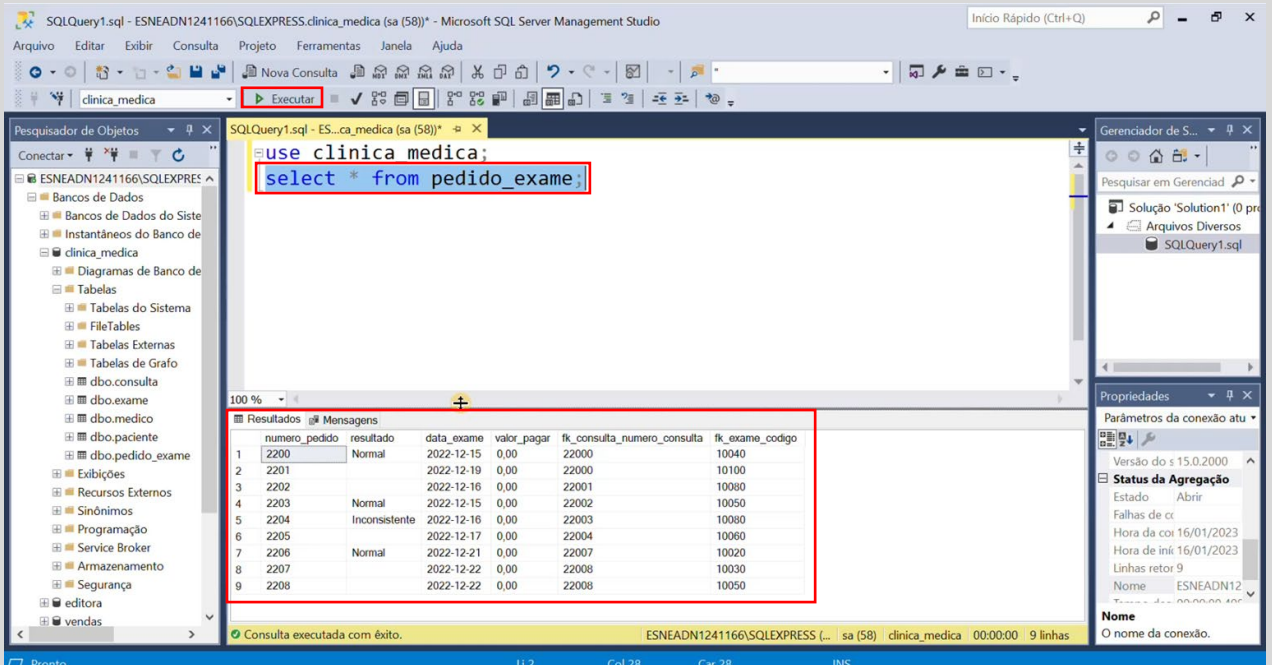
1. Abra o SSMS e clique em **Nova Consulta**.



2. Digite **use clinica\_medica;** e clique em **Executar** para selecionar a base de dados que será usada. Você pode usar o **menu** ao lado do botão Executar também.



3. Digite **select \* from pedido\_exame;** e clique em **Executar** para mostrar todos os dados da tabela **pedido\_exame**.



The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
use clinica medica;
select * from pedido_exame;
```

The 'Executar' button is highlighted in red. The results pane displays the following data:

numero_pedido	resultado	data_exame	valor_pagar	fk_consulta_numero_consulta	fk_exame_codigo
2200	Normal	2022-12-15	0,00	22000	10040
2201		2022-12-19	0,00	22000	10100
2202		2022-12-16	0,00	22001	10080
2203	Normal	2022-12-15	0,00	22002	10050
2204	Inconsistente	2022-12-16	0,00	22003	10080
2205		2022-12-17	0,00	22004	10060
2206	Normal	2022-12-21	0,00	22007	10020
2207		2022-12-22	0,00	22008	10030
2208		2022-12-22	0,00	22008	10050

The status bar at the bottom indicates: 'Consulta executada com êxito. ESNEADN1241166\SQLEXPRESS (sa (58)) clinica\_medica 00:00:00 9 linhas'.

Note que a coluna **valor\_pagar** está zerada, pois os dados de exemplo foram inseridos antes da criação da *trigger* de cálculo. Antes de criar o gatilho, devemos deletar os dados inseridos, pois o gatilho será disparado pela inserção de dados.

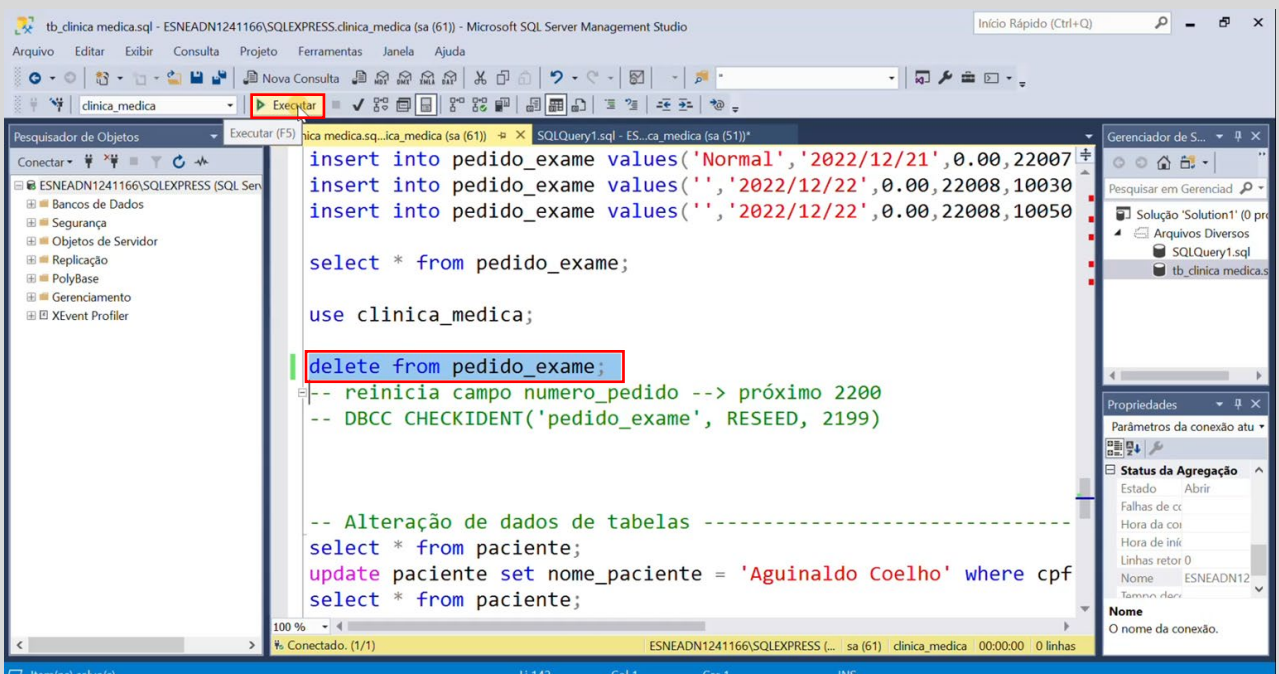
### Dica!

As funcionalidades das triggers, como calcular automaticamente o valor a pagar, podem ser feitas através de aplicações externas ao banco de dados. A vantagem de usar triggers é a rapidez.



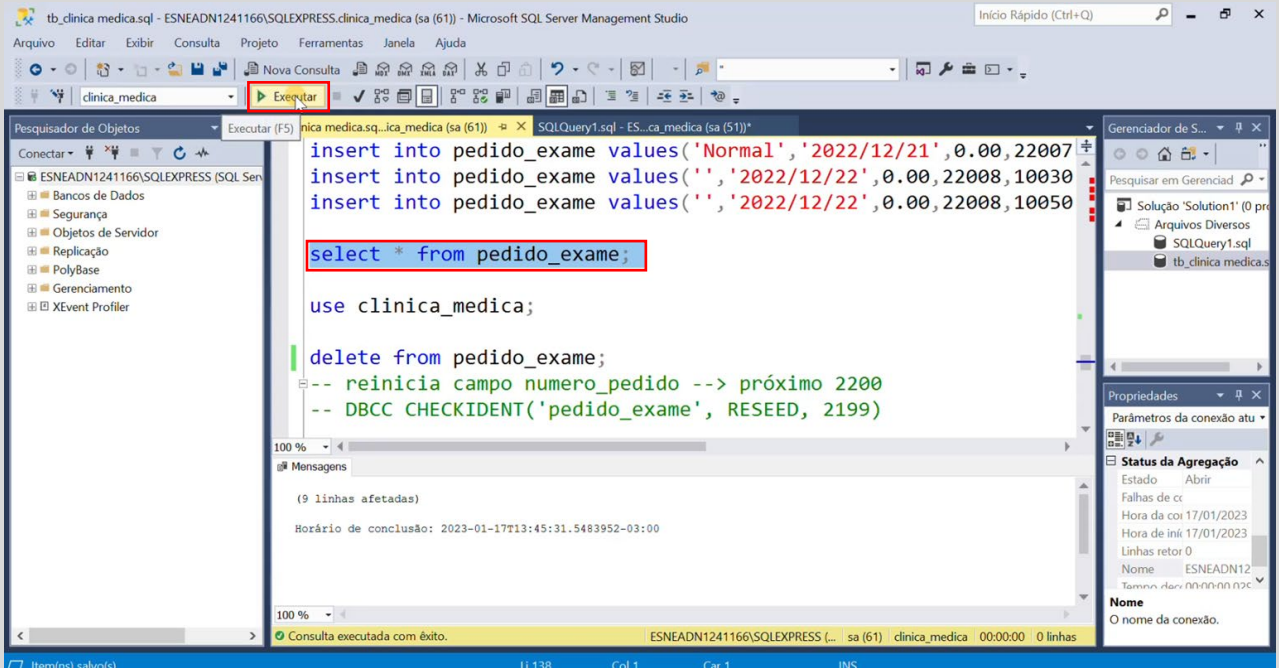
5. Para apagar os dados da tabela, abra o script de comandos ddl e dml, nomeado como **tb\_clinica\_medica.sql**. Nesse arquivo sql estão alguns comandos úteis para a atividade. Você pode carregar o script todo e selecionar apenas o comando que quiser executar.

Encontre o comando **delete from pedido\_exame;**, retire o comentário (os dois traços -- ), então selecione o comando e clique em **Executar**.

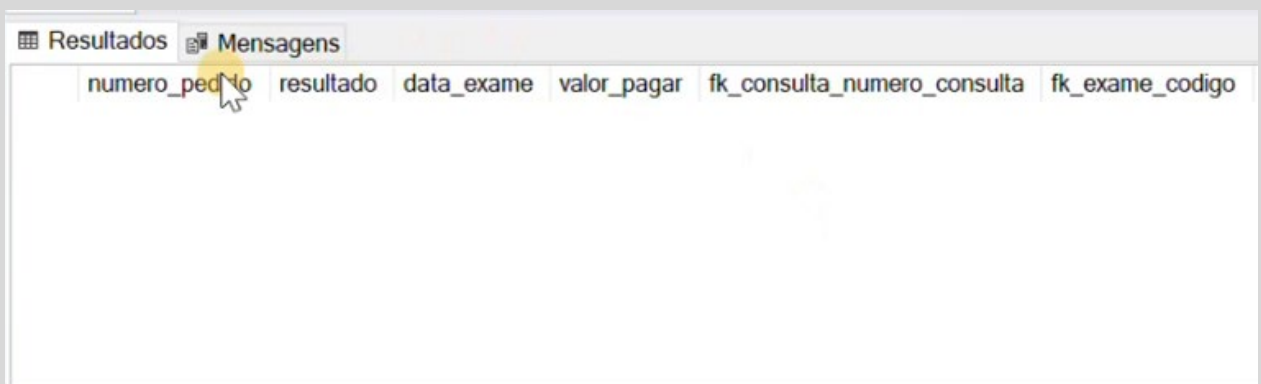


Você pode também apenas digitar **delete from pedido\_exame;** na janela que estiver usando, selecionar o comando e clicar em **Executar**.

6. Para verificar se os dados foram apagados, selecione **select \* from pedido\_exame** e clique em **Executar**.



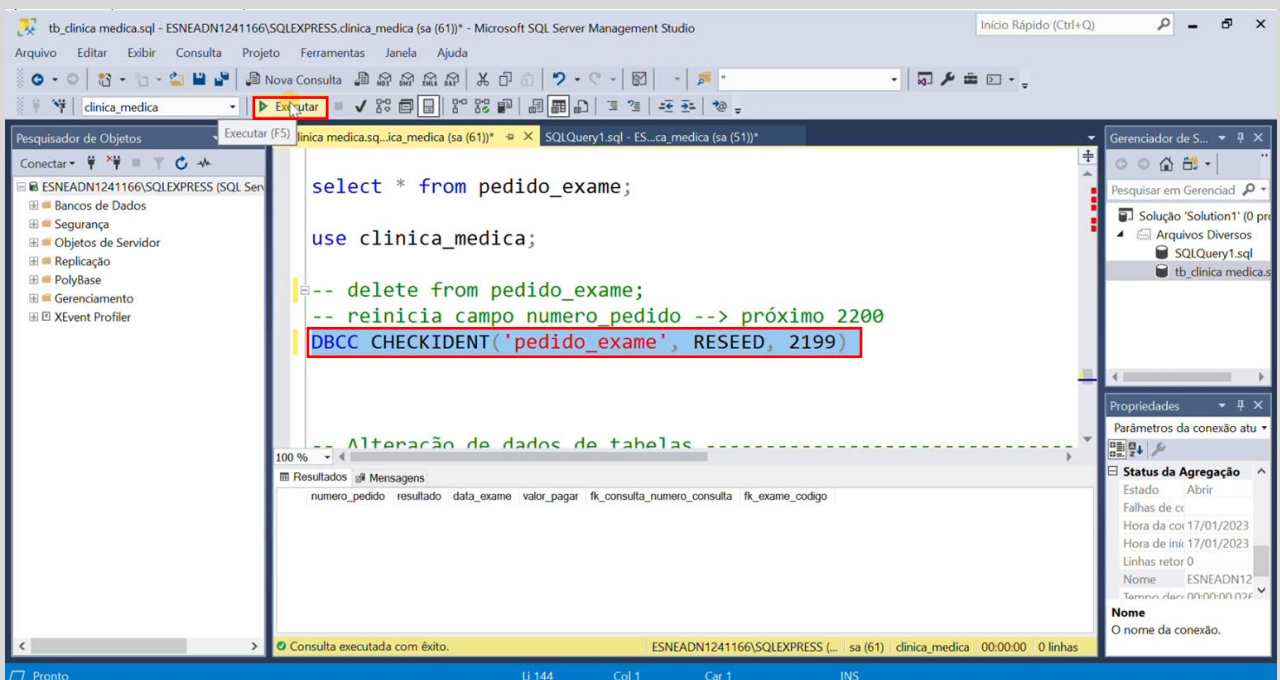
No terminal do SSMS, a tabela pedido\_exame estará sem dados.





7. Note que cada um dos nove exames deletados tinha um número, de 2200 a 2208. O próximo pedido de exame que for inserido agora será o 2209.

Para essa atividade, vamos inserir os dados reiniciando a contagem em 2200. Encontre (ou digite) o comando **DBCC CHECKIDENT('pedido\_exame', RESEED, 2199)**, selecione-o e clique em **Executar**.



## 8. A estrutura básica da trigger é:

```
create trigger Nome_da_trigger
    on tabela_de_referencia
    after ação
    as
begin
    -- procedimentos dentro da trigger
end
```

Na nossa atividade, a estrutura básica fica:

```
create trigger Atualiza_Pedido_Exame
    on pedido_exame
    after insert
    as
begin
    -- procedimentos dentro da trigger
end
```

9. Entre os comando begin e end, vamos declarar as variáveis com base no modelo lógico e programar as ações da trigger.

No bloco a seguir, definimos que a contagem não será retornada para reduzir tarefas. Declaramos os tipos e os nomes das variáveis para armazenar os valores inseridos para número do pedido de exame, número da consulta, código e preço do exame.

```
SET NOCOUNT ON
declare @num_ped as integer;
select @num_ped = numero_pedido from inserted;
declare @num_cons as integer;
select @num_cons = fk_consulta_numero_consulta from inserted;
declare @cod_ex as integer;
select @cod_ex = fk_exame_codigo from inserted;
declare @prc as money;
```

10. No bloco seguinte, selecionamos o preço do exame pelo código. Declaramos o tipo e o nome da variável para armazenar o CPF do paciente. Selecionamos então o CPF do paciente pelo número da consulta. Selecionamos também o tipo do plano do paciente pelo CPF.

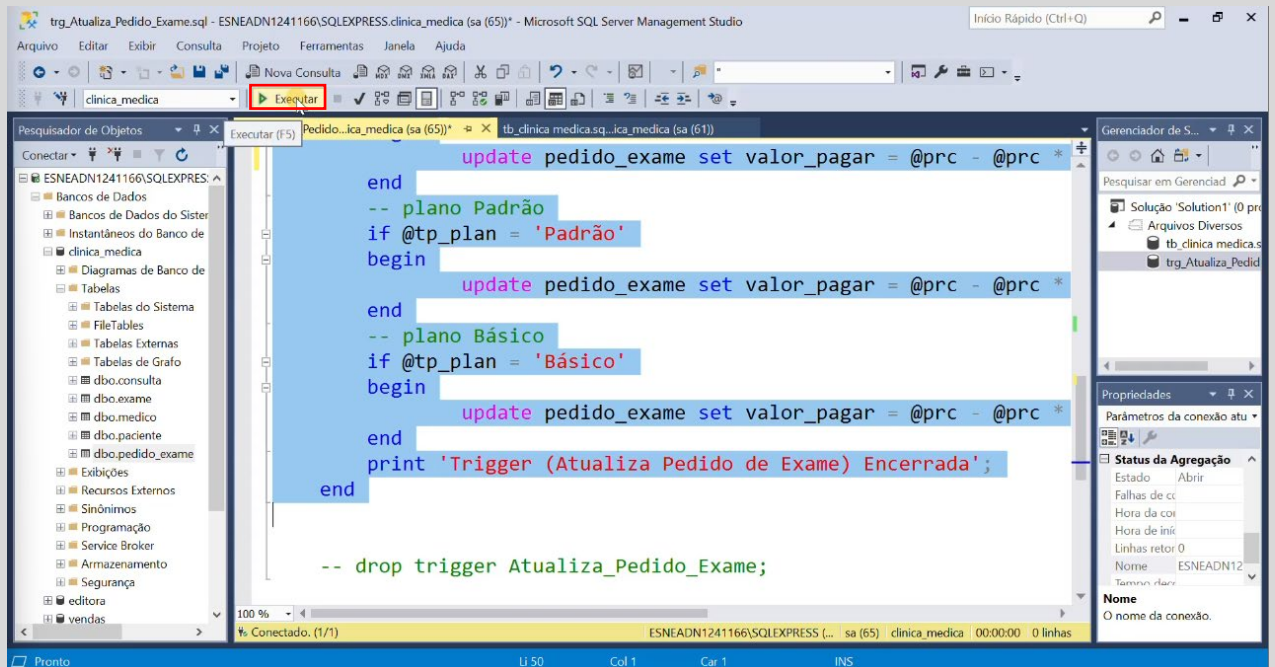
```
select @prc = preco from exame where codigo = @cod_ex;
declare @cpf_pac as varchar(20);
select @cpf_pac = fk_paciente_cpf from consulta where numero_consulta = @num_cons;
declare @tp_plan as varchar(20);
select @tp_plan = tipo_plano from paciente where cpf = @cpf_pac;
```

11. No bloco a seguir, definimos uma estrutura condicional, de acordo com o tipo de plano. Depois atualizamos o valor a pagar pelo exame, já com o desconto de acordo com o plano. Por fim, enviaremos uma mensagem de trigger encerrada.

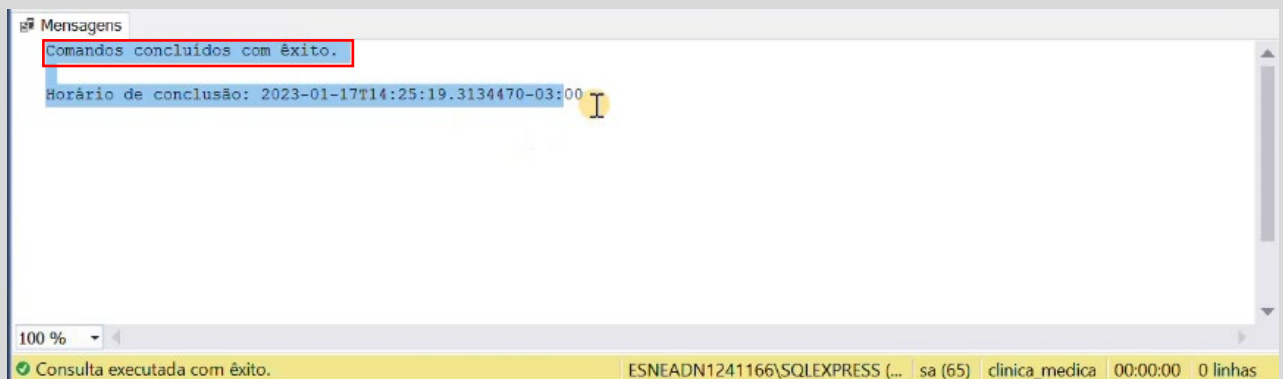
```
if @tp_plan = 'Especial'
begin
    update pedido_exame set valor_pagar = @prc - @prc * 100 / 100 where
numero_pedido = @num_ped;
end
if @tp_plan = 'Padrão'
begin
    update pedido_exame set valor_pagar = @prc - @prc * 30 / 100 where
numero_pedido = @num_ped;
end
if @tp_plan = 'Básico'
begin
    update pedido_exame set valor_pagar = @prc - @prc * 10 / 100 where
numero_pedido = @num_ped;
end
print 'Trigger (Atualiza Pedido de Exame) Encerrada';
end
```



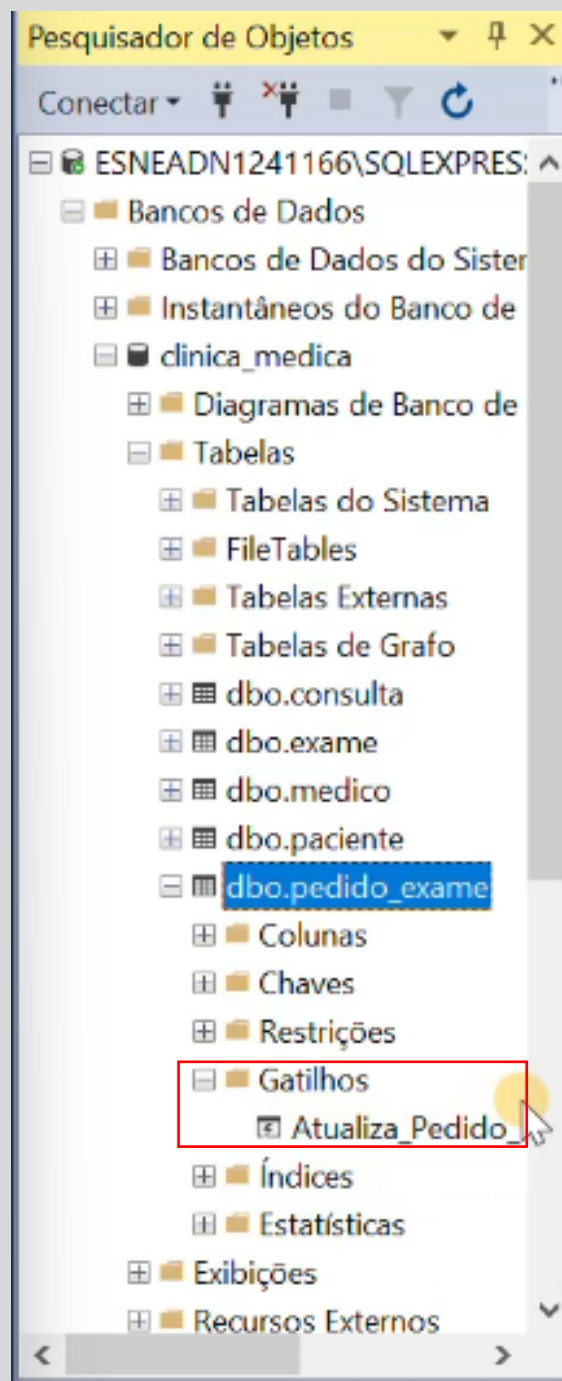
## 12. Selecione todo o script e clique em Executar.



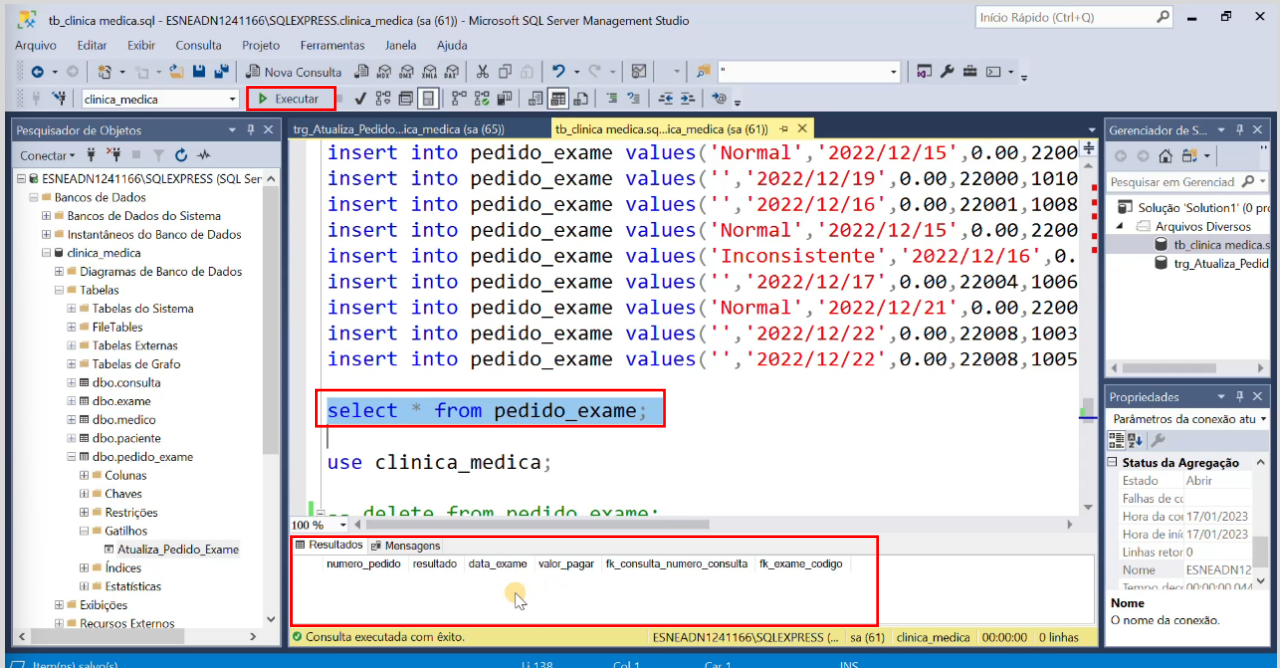
No terminal, deve aparecer uma mensagem de êxito.



12. Depois de executada uma vez, sua trigger ficará armazenada dentro da pasta Gatilhos, dentro da tabela de referência.



13. Selecione o comando `select * from pedido_exame;` e clique em Executar. A tabela estará sem dados: a trigger só atualizará os valores a serem pagos depois que os dados forem inseridos.



14. Selecione o bloco de inserção de dados a seguir no arquivo `tb_clinica medica.sql` e clique em Executar. O terminal deve mostrar um alerta para cada linha atualizada pela trigger.

```
insert into pedido_exame values('Normal','2022/12/15',0.00,22000,1
insert into pedido_exame values('','2022/12/19',0.00,22000,10100);
insert into pedido_exame values('','2022/12/16',0.00,22001,10080);
insert into pedido_exame values('Normal','2022/12/15',0.00,22002,1
insert into pedido_exame values('Inconsistente','2022/12/16',0.00,
insert into pedido_exame values('','2022/12/17',0.00,22004,10060);
insert into pedido_exame values('Normal','2022/12/21',0.00,22007,1
insert into pedido_exame values('','2022/12/22',0.00,22008,10030);
insert into pedido_exame values('','2022/12/22',0.00,22008,10050);
```

15. Selecione o comando `select * from pedido_exame;` e clique em Executar. Note que a contagem do pedido recomeçou em 2200 e a coluna do valor a pagar já está atualizada.

The screenshot shows the Microsoft SQL Server Management Studio interface. The query editor contains the following SQL code:

```
insert into pedido_exame values('Inconsistente','2022/12/16',0.00,22004,1006)
insert into pedido_exame values('','2022/12/17',0.00,22004,1006)
insert into pedido_exame values('Normal','2022/12/21',0.00,22008,1003)
insert into pedido_exame values('','2022/12/22',0.00,22008,1005)

select * from pedido_exame;
```

The 'Executar' button is highlighted. The results grid shows the following data:

numero_pedido	resultado	data_exame	valor_pagar	fk_consulta_numero_consulta	fk_exame_codigo
2200	Normal	2022-12-15	385,00	22000	10040
2201		2022-12-19	35,00	22000	10100
2202		2022-12-16	270,00	22001	10080
2203	Normal	2022-12-15	720,00	22002	10050
2204	Inconsistente	2022-12-16	0,00	22003	10080
2205		2022-12-17	49,00	22004	10060
2206	Normal	2022-12-21	70,00	22007	10020
2207		2022-12-22	175,00	22008	10030
2208		2022-12-22	560,00	22008	10050

The status bar at the bottom indicates: 'Consulta executada com êxito. ESNEADN1241166\SQLEXPRESS [sa (61)] clinica\_medica : 00:00:00 9 linhas'.

## Procedimentos armazenados (stored procedures)

Nessa próxima etapa, você vai acompanhar como criar procedimentos armazenados para a consulta de:

- Agenda de médicos
- Exames solicitados
- Histórico de pagamentos
- Resumo de pagamentos

A estrutura básica de uma store procedure é:

```
create procedure Nome_Procedure
as
begin
    -- comandos sql
end

execute Nome_Procedure
```

### Importante

A principal diferença entre uma trigger e uma stored procedure é que a trigger é disparada automaticamente por uma ação pré-configurada e o store procedure necessita de um comando para ser executada.



1. Na stored procedure **Agenda\_Medicos**, vamos selecionar dados de várias tabelas diferentes. Com base no modelo lógico, podemos indicar corretamente quais os campos que desejamos selecionar e relacionar.

```
create procedure Agenda_Medicos
as
begin
    select m.nome_medico, m.especialidade, m.crm, c.numero_consulta,
           c.data_consulta, c.horario_consulta, p.nome_paciente, p.cpf,
           p.nome_plano, p.tipo_plano from medico as m inner join consulta as c
           on m.crm = c.fk_medico_crm inner join paciente as p on
           c.fk_paciente_cpf = p.cpf
           order by m.nome_medico, c.data_consulta;
end

execute Agenda_Medicos;
```

Os dados selecionados aparecerão em ordem alfabética pelo nome do médico e por data da consulta.

### Dica!

Você pode dar um “apelido” para cada tabela com a cláusula as:

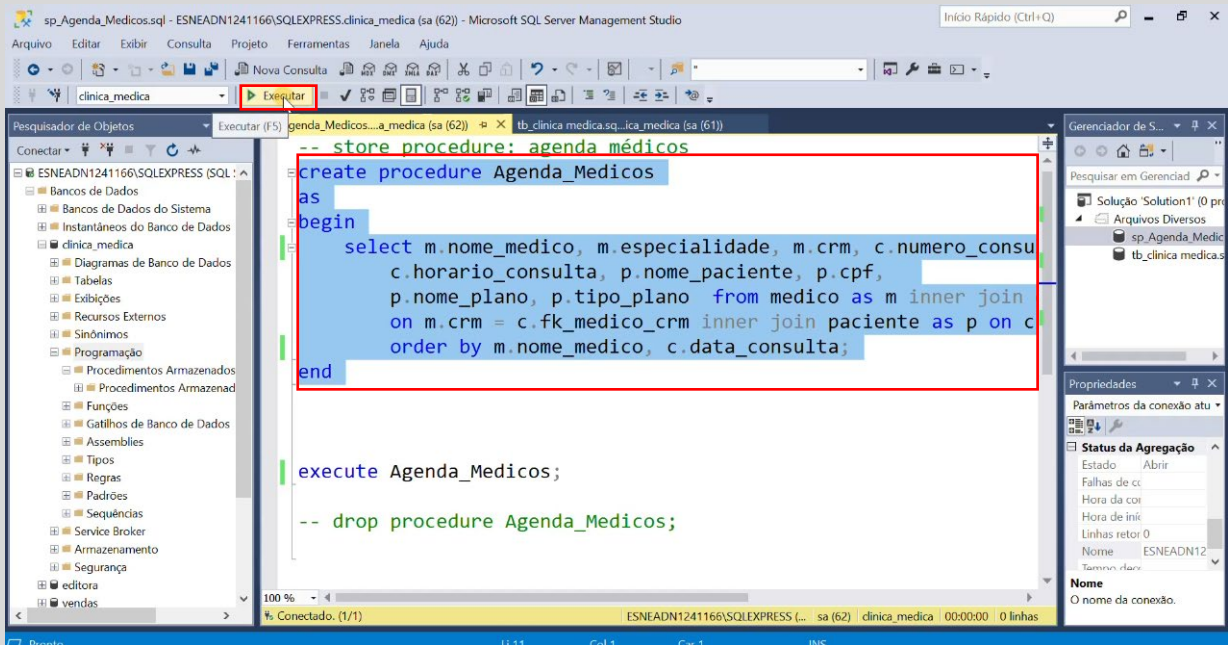
**medico as m**

Assim m.nome\_medico seria o nome do médico dentro da tabela medico.

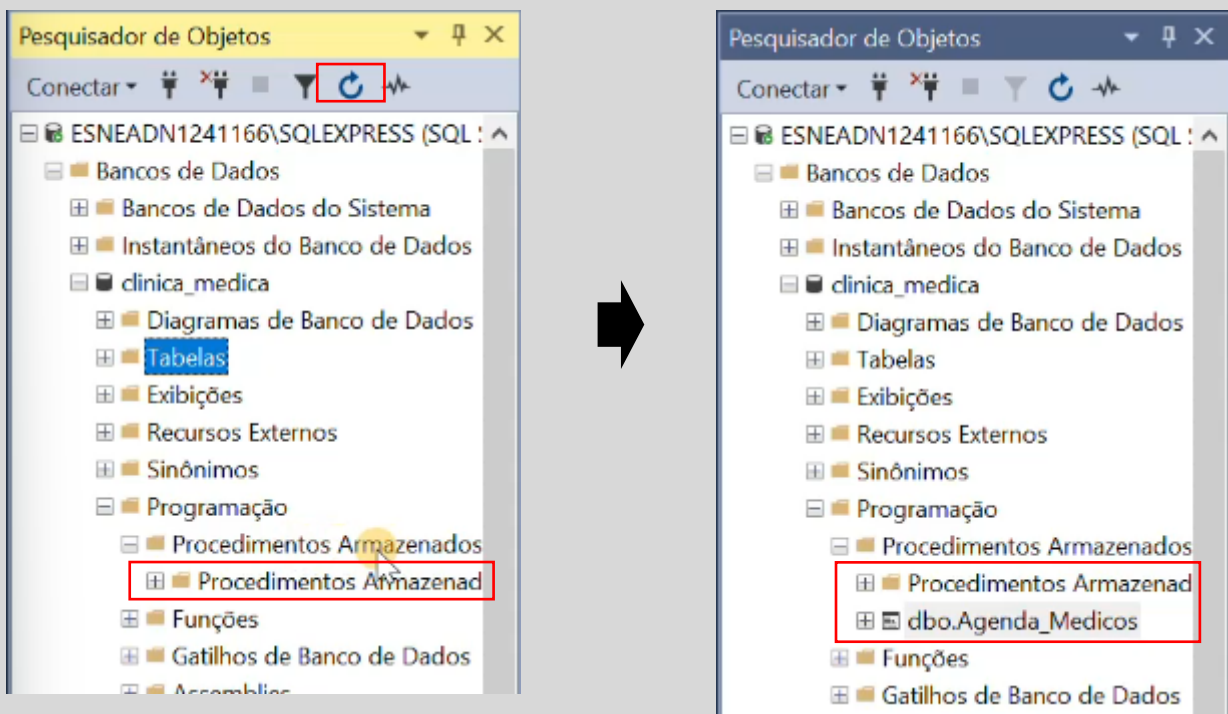




## 2. Selecione a store procedure **Agenda\_Medicos** e clique em **Executar**.



Clique em **Atualizar** e note que o procedimento foi armazenado na pasta Procedimentos Armazenados.



3. Agora selecione o comando **execute Agenda\_Medicos** e clique em **Executar**. Note que no terminal os dados selecionados foram listados em ordem alfabética pelo nome do médico e por dia de consulta.

The screenshot shows the SSMS interface with the following components:

- Menu Bar:** Arquivo, Editar, Exibir, Consulta, Projeto, Ferramentas, Janela, Ajuda.
- Toolbar:** Includes buttons for 'Nova Consulta', 'Executar' (highlighted with a red box), and other standard SSMS icons.
- Left Pane (Pesquisador de Objetos):** Shows the database structure for 'ESNEADN1241166\SQLEXPRESS (SQL S...)' with folders for 'Bancos de Dados do Sistema', 'Instantâneos do Banco de Dados', 'clínica\_medica', 'Diagramas de Banco de Dados', 'Tabelas', 'Exibições', 'Recursos Externos', 'Sinônimos', 'Programação', 'Procedimentos Armazenados', 'Funções', 'Gatilhos de Banco de Dados', 'Assemblies', 'Tipos', 'Regras', 'Padrões', 'Sequências', 'Service Broker', 'Armazenamento', and 'Segurança'.
- Central Query Editor:** Contains the SQL script:
 

```

      p.nome_plano, p.tipo_plano from medico as m inner join
      on m.crm = c.fk_medico_crm inner join paciente as p on c
      order by m.nome_medico, c.data_consulta;

      end

      execute Agenda_Medicos;

      -- drop procedure Agenda_Medicos;
      
```
- Bottom Pane (Resultados):** Displays the results of the query in a table format. The table has columns: nome\_medico, especialidade, crm, numero\_consulta, data\_consulta, horario\_consulta, nome\_paciente, cpf, nome\_plano, and tipo\_plano. The results are sorted alphabetically by doctor name and then by consultation date.
 

nome_medico	especialidade	crm	numero_consulta	data_consulta	horario_consulta	nome_paciente	cpf	nome_plano	tipo_plano
Agildo Nunes	Cardiologia	102030	22000	2022-12-12	14:30:00.0000000	Leonardo Ribeiro	012.345.678-90	Inovamed	Padrão
Agildo Nunes	Cardiologia	102030	22006	2022-12-19	16:45:00.0000000	Gilberto Barros	234.567.890-23	Inovamed	Especial
Agildo Nunes	Cardiologia	102030	22009	2022-12-20	14:20:00.0000000	Bruna Alvez	123.456.789-12	Ultramed	Básico
Edna Cardoso	Ortopedia	405060	22004	2022-12-15	15:00:00.0000000	Maria Pereira	345.678.901-45	Ultramed	Padrão
Edna Cardoso	Ortopedia	405060	22007	2022-12-16	18:00:00.0000000	Maria Pereira	345.678.901-45	Ultramed	Padrão
- Status Bar:** Shows 'Pronto', 'Linha 15', 'Col. 1', 'Car. 1', and 'INS'.

4. Para os procedimentos Exames\_Solicitados e Historico\_Pagamentos, você deve repetir os passos feitos para Agenda\_Medicos:

- Abrir o script no SSMS
- Selecionar todo o script
- Clicar em Executar
- Verificar se o procedimento foi armazenado e executado com êxito

Confira a seguir o resultado da execução das stored procedures Agenda\_Medicos, Exames\_Solicitados e Historico\_Pagamentos

### Agenda\_Medicos

	nome_medico	especialidade	crm	numero_consulta	data_consulta	horario_consulta	nome_paciente	cpf	nome_plano	tipo_plano
1	Agildo Nunes	Cardiologia	102030	22000	2022-12-12	14:30:00.0000000	Leonardo Ribeiro	012.345.678-90	Inovamed	Padrão
2	Agildo Nunes	Cardiologia	102030	22006	2022-12-19	16:45:00.0000000	Gilberto Barros	234.567.890-23	Inovamed	Especial
3	Agildo Nunes	Cardiologia	102030	22009	2022-12-20	14:20:00.0000000	Bruna Alvez	123.456.789-12	Ultramed	Básico
4	Edna Cardoso	Ortopedia	405060	22004	2022-12-15	15:00:00.0000000	Maria Pereira	345.678.901-45	Ultramed	Padrão
5	Lúcia Marques	Pediatria	607080	22007	2022-12-19	18:00:00.0000000	Maria Pereira	345.678.901-45	Ultramed	Padrão
6	Márcia Alvez	Gastrologia	203040	22001	2022-12-13	08:30:00.0000000	Bruna Alvez	123.456.789-12	Ultramed	Básico
7	Márcia Alvez	Gastrologia	203040	22003	2022-12-14	10:00:00.0000000	Gilberto Barros	234.567.890-23	Inovamed	Especial
8	Ricardo Souza	Otorrinolaring...	506070	22005	2022-12-16	10:00:00.0000000	Bruna Alvez	123.456.789-12	Ultramed	Básico
9	Roberto Gus...	Neurologia	304050	22002	2022-12-13	11:00:00.0000000	Bruna Alvez	123.456.789-12	Ultramed	Básico
10	Roberto Gus...	Neurologia	304050	22008	2022-12-20	09:00:00.0000000	Leonardo Ribeiro	012.345.678-90	Inovamed	Padrão

### Exames\_Solicitados

	nome_medico	especialidade	crm	numero_consulta	numero_pedido	data_exame	codigo	especificacao
1	Agildo Nunes	Cardiologia	102030	22000	2200	2022-12-15	10040	Ultrassonografia
2	Agildo Nunes	Cardiologia	102030	22000	2201	2022-12-19	10100	Eletrocardiograma
3	Edna Cardoso	Ortopedia	405060	22004	2205	2022-12-17	10060	Radiografia
4	Lúcia Marques	Pediatria	607080	22007	2206	2022-12-21	10020	Hemograma
5	Márcia Alvez	Gastrologia	203040	22001	2202	2022-12-16	10080	Endoscopia
6	Márcia Alvez	Gastrologia	203040	22003	2204	2022-12-16	10080	Endoscopia
7	Roberto Gusmão	Neurologia	304050	22002	2203	2022-12-15	10050	Ressonância
8	Roberto Gusmão	Neurologia	304050	22008	2207	2022-12-22	10030	Tomografia
9	Roberto Gusmão	Neurologia	304050	22008	2208	2022-12-22	10050	Ressonância

### Historico\_Pagamentos

	nome_paciente	cpf	numero_consulta	data_consulta	data_exame	valor_pagar	codigo	especificacao
1	Bruna Alvez	123.456.789-12	22002	2022-12-13	2022-12-15	720,00	10050	Ressonância
2	Bruna Alvez	123.456.789-12	22001	2022-12-13	2022-12-16	270,00	10080	Endoscopia
3	Gilberto Barros	234.567.890-23	22003	2022-12-14	2022-12-16	0,00	10080	Endoscopia
4	Leonardo Ribeiro	012.345.678-90	22000	2022-12-12	2022-12-15	385,00	10040	Ultrassonografia
5	Leonardo Ribeiro	012.345.678-90	22000	2022-12-12	2022-12-19	35,00	10100	Eletrocardiograma
6	Leonardo Ribeiro	012.345.678-90	22008	2022-12-20	2022-12-22	175,00	10030	Tomografia
7	Leonardo Ribeiro	012.345.678-90	22008	2022-12-20	2022-12-22	560,00	10050	Ressonância
8	Maria Pereira	345.678.901-45	22004	2022-12-15	2022-12-17	49,00	10060	Radiografia
9	Maria Pereira	345.678.901-45	22007	2022-12-19	2022-12-21	70,00	10020	Hemograma

5. Para o resumo de pagamentos, vamos detalhar uma diferença: o uso de parâmetros. No código a seguir, note que na criação do procedimento há um parâmetro, cujo valor será buscado na coluna nome\_pac e deve ser uma varchar de até 40 caracteres.

```
create procedure Resumo_Pagamentos @nome_pac varchar(40)
as
Begin
select pa.nome_paciente, sum(pe.valor_pagar) as total_pagar
    from paciente as pa inner join consulta as c
    on pa.cpf = c.fk_paciente_cpf inner join pedido_exame as pe
    on c.numero_consulta = pe.fk_consulta_numero_consulta
    where pa.nome_paciente = @nome_pac
    group by pa.nome_paciente;
end

execute Resumo_Pagamentos 'Leonardo Ribeiro';
```

No exemplo, o procedimento recebeu o parâmetro Leonardo Ribeiro, então serão selecionados os dados do paciente com esse nome.

### Dica!

O parâmetro deve estar exatamente igual ao valor da coluna.



6. Selecione toda a **stored procedure** e clique em **Executar**. O resultado aparecerá no terminal.

	nome_paciente	(Nenhum nome de coluna)
1	Leonardo Ribeiro	1155,00

Experimente trocar o parâmetro para **Maria Pereira**. Então selecione o comando **execute Resumo\_Pagamentos Maria Pereira**; e clique em **Executar**. O resultado no terminal será:

	nome_paciente	(Nenhum nome de coluna)
1	Maria Pereira	119,00

## Scripts (códigos)

Confira a seguir os scripts usados para essa atividade.

### tb\_clinica medica.sql

```
-- drop database clinica_medica;

create database clinica_medica;

use clinica_medica;

CREATE TABLE paciente (
    cpf varchar(14) PRIMARY KEY,
    nome_paciente varchar(40),
    telefone varchar(14),
    numero_plano int,
    nome_plano varchar(20),
    tipo_plano varchar(10)
);

CREATE TABLE medico (
    crm int PRIMARY KEY,
    nome_medico varchar(30),
    especialidade varchar(20)
);

-- auto incremento --
-- identity (valor inicial, incremento) --
CREATE TABLE pedido_exame (
    numero_pedido int identity(2200,1) PRIMARY KEY,
    resultado varchar(40),
    data_exame date,
    valor_pagar money,
    fk_consulta_numero_consulta int,
    fk_exame_codigo int
);
```



```
-- auto incremento --
-- identity (valor inicial, incremento) --
CREATE TABLE consulta (
    numero_consulta int identity(22000,1) PRIMARY KEY,
    data_consulta date,
    horario_consulta time,
    fk_paciente_cpf varchar(14),
    fk_medico_crm int
);

CREATE TABLE exame (
    codigo int PRIMARY KEY,
    especificacao varchar(20),
    preco money
);

ALTER TABLE pedido_exame ADD CONSTRAINT FK_pedido_exame_2
    FOREIGN KEY (fk_consulta_numero_consulta)
    REFERENCES consulta (numero_consulta)
    ON DELETE CASCADE;

ALTER TABLE pedido_exame ADD CONSTRAINT FK_pedido_exame_3
    FOREIGN KEY (fk_exame_codigo)
    REFERENCES exame (codigo);

ALTER TABLE consulta ADD CONSTRAINT FK_consulta_2
    FOREIGN KEY (fk_paciente_cpf)
    REFERENCES paciente (cpf)
    ON DELETE CASCADE;

ALTER TABLE consulta ADD CONSTRAINT FK_consulta_3
    FOREIGN KEY (fk_medico_crm)
    REFERENCES medico (crm)
    ON DELETE CASCADE;
```

```
-- inclusão e seleção de dados -----
-----

-- tabela paciente
insert into paciente values('012.345.678-90','Leonardo
Ribeiro','(11)91234-5678',123456,'Inovamed','Padrão');
insert into paciente values('123.456.789-12','Bruna
Alvez','(15)92345-6789',234567,'Ultramed','Básico');
insert into paciente values('234.567.890-23','Gilberto
Barros','(11)94567-8901',345678,'Inovamed','Especial');
insert into paciente values('345.678.901-45','Maria
Pereira','(12)95678-9012',456789,'Ultramed','Padrão');
insert into paciente values('456.789.012-34','Arnaldo
Coelho','(19)96789-0123',567890,'Inovamed','Especial');

select * from paciente;

-- tabela medico
insert into medico values(102030,'Agildo Nunes','Cardiologia');
insert into medico values(203040,'Márcia Alvez','Gastrologia');
insert into medico values(304050,'Roberto Gusmão','Neurologia');
insert into medico values(405060,'Edna Cardoso','Ortopedia');
insert into medico values(506070,'Ricardo
Souza','Otorrinolaringologia');
insert into medico values(607080,'Lúcia Marques','Pediatria');
insert into medico values(708090,'Beatriz Lucena','Oncologia');

select * from medico;

-- tabela exame
insert into exame values(10020,'Hemograma',100.00);
insert into exame values(10030,'Tomografia',250.00);
insert into exame values(10040,'Ultrassonografia',550.00);
insert into exame values(10050,'Ressonância',800.00);
insert into exame values(10060,'Radiografia',70.00);
insert into exame values(10070,'Mamografia',150.00);
insert into exame values(10080,'Endoscopia',300.00);
insert into exame values(10090,'Colonoscopia',300.00);
insert into exame values(10100,'Eletrocardiograma',50.00);
insert into exame values(10110,'Ecocardiograma',120.00);
insert into exame values(10120,'Audiometria',65.00);
```

```

select * from exame;

-- campos 'auto incremento' não devem aparecer no insert
-- tabela consulta
insert into consulta values('2022/12/12','14:30','012.345.678-90',102030);
insert into consulta values('2022/12/13','08:30','123.456.789-12',203040);
insert into consulta values('2022/12/13','11:00','123.456.789-12',304050);
insert into consulta values('2022/12/14','10:00','234.567.890-23',203040);
insert into consulta values('2022/12/15','15:00','345.678.901-45',405060);
insert into consulta values('2022/12/16','10:00','123.456.789-12',506070);
insert into consulta values('2022/12/19','16:45','234.567.890-23',102030);
insert into consulta values('2022/12/19','18:00','345.678.901-45',607080);
insert into consulta values('2022/12/20','09:00','012.345.678-90',304050);
insert into consulta values('2022/12/20','14:20','123.456.789-12',102030);

select * from consulta;

-- campo 'auto incremento' não deve aparecer no insert
-- resultados possíveis: (normal, inconsistente, indeterminado, negativo, positivo)
insert into pedido_exame
values('Normal','2022/12/15',0.00,22000,10040);
insert into pedido_exame values('','2022/12/19',0.00,22000,10100);
insert into pedido_exame values('','2022/12/16',0.00,22001,10080);
insert into pedido_exame
values('Normal','2022/12/15',0.00,22002,10050);
insert into pedido_exame
values('Inconsistente','2022/12/16',0.00,22003,10080);
insert into pedido_exame values('','2022/12/17',0.00,22004,10060);
insert into pedido_exame
values('Normal','2022/12/21',0.00,22007,10020);
insert into pedido_exame values('','2022/12/22',0.00,22008,10030);
insert into pedido_exame values('','2022/12/22',0.00,22008,10050);

```

```
select * from pedido_exame;

use clinica_medica;

-- delete from pedido_exame;

-- reinicia campo numero_pedido --> próximo 2200
-- DBCC CHECKIDENT('pedido_exame', RESEED, 2199)

-- Alteração de dados de tabelas -----
select * from paciente;
update paciente set nome_paciente = 'Aguinaldo Coelho' where cpf =
'456.789.012-34';
select * from paciente;

select * from medico;
update medico set especialidade = 'Ginecologia' where crm = 708090;
select * from medico;

select * from exame;
update exame set preco = 135.00 where codigo = 10110;
select * from exame;

-- Exclusão de registros de tabelas -----

select * from paciente;
delete from paciente where cpf = '456.789.012-34';
select * from paciente;

select * from medico;
delete from medico where crm = 708090;
select * from medico;
```

**trg\_Atualiza\_Pedido\_Exame.sql**

```

-- gatilho de cálculo de valor a pagar da tabela pedido_exame
create trigger Atualiza_Pedido_Exame
  on pedido_exame
  after insert
  as
  begin
    SET NOCOUNT ON
    declare @num_ped as integer;
    -- captura numero do pedido do registro inserido
    select @num_ped = numero_pedido from inserted;
    declare @num_cons as integer;
    -- captura numero da consulta do registro inserido
    select @num_cons = fk_consulta_numero_consulta from inserted;
    declare @cod_ex as integer;
    -- captura codigo exame do registro inserido
    select @cod_ex = fk_exame_codigo from inserted;
    declare @prc as money;
    -- busca preço do exame
    select @prc = preco from exame where codigo = @cod_ex;
    declare @cpf_pac as varchar(20);
    -- busca CPF paciente
    select @cpf_pac = fk_paciente_cpf from consulta where
numero_consulta = @num_cons;
    declare @tp_plan as varchar(20);
    -- busca tipo de plano de saúde do paciente
    select @tp_plan = tipo_plano from paciente where cpf =
@cpf_pac;
    -- print 'Número Pedido: ' + convert(varchar(30),@num_ped);
    -- print 'Número Consulta: ' +
convert(varchar(30),@num_cons);
    -- print 'Código Exame: ' + convert(varchar(30),@cod_ex);
    -- print 'Preço Exame: ' + convert(varchar(30),@prc);
    -- print 'CPF Paciente: ' + @cpf_pac;
    -- print 'Tipo Plano: ' + @tp_plan;

```

```
-- plano Especial
if @tp_plan = 'Especial'
begin
    update pedido_exame set valor_pagar = @prc - @prc *
100 / 100 where numero_pedido = @num_ped;
end
-- plano Padrão
if @tp_plan = 'Padrão'
begin
    update pedido_exame set valor_pagar = @prc - @prc *
30 / 100 where numero_pedido = @num_ped;
end
-- plano Básico
if @tp_plan = 'Básico'
begin
    update pedido_exame set valor_pagar = @prc - @prc *
10 / 100 where numero_pedido = @num_ped;
end
print 'Trigger (Atualiza Pedido de Exame) Encerrada';
end

-- drop trigger Atualiza_Pedido_Exame;
```



## sp\_Agenda\_Medicos.sql

```
-- store procedure: agenda médicos
create procedure Agenda_Medicos
as
begin
    select m.nome_medico, m.especialidade, m.crm, c.numero_consulta,
    c.data_consulta,
        c.horario_consulta, p.nome_paciente, p.cpf,
        p.nome_plano, p.tipo_plano  from medico as m inner join
consulta as c
        on m.crm = c.fk_medico_crm inner join paciente as p on
c.fk_paciente_cpf = p.cpf
        order by m.nome_medico, c.data_consulta;
end

execute Agenda_Medicos;

-- drop procedure Agenda_Medicos;
```

## sp\_Exames\_Solicitados.sql

```
-- store procedure: exames solicitados em ordem de médico
create procedure Exames_Solicitados
as
begin
select m.nome_medico, m.especialidade, m.crm, c.numero_consulta,
       p.numero_pedido, p.data_exame, e.codigo, e.especificacao
  from medico as m inner join consulta as c
    on m.crm = c.fk_medico_crm inner join pedido_exame as p
    on c.numero_consulta = p.fk_consulta_numero_consulta
    inner join exame as e on p.fk_exame_codigo = e.codigo
 order by m.nome_medico, p.data_exame;
end

execute Exames_Solicitados;

-- drop procedure Exames_Solicitados;
```

## sp\_Historico\_Pagamentos.sql

```
-- store procedure: histórico pagamentos dos pacientes
create procedure Historico_Pagamentos
as
begin
select pa.nome_paciente, pa.cpf, c.numero_consulta,
c.data_consulta,
    pe.data_exame, pe.valor_pagar,
    e.codigo, e.especificacao from paciente as pa inner join
consulta as c
    on pa.cpf = c.fk_paciente_cpf inner join pedido_exame as pe
    on c.numero_consulta = pe.fk_consulta_numero_consulta inner
join exame as e
    on pe.fk_exame_codigo = e.codigo
    order by pa.nome_paciente, pe.data_exame;
end

execute Historico_Pagamentos;

-- drop procedure Historico_Pagamentos;
```

## sp\_Resumo\_Pagamentos.sql

```
-- store procedure: resumo pagamentos por paciente
create procedure Resumo_Pagamentos @nome_pac varchar(40)
as
begin
select pa.nome_paciente, sum(pe.valor_pagar) as total_pagar
      from paciente as pa inner join consulta as c
      on pa.cpf = c.fk_paciente_cpf inner join pedido_exame as pe
      on c.numero_consulta = pe.fk_consulta_numero_consulta
      where pa.nome_paciente = @nome_pac
      group by pa.nome_paciente;
end

execute Resumo_Pagamentos 'Leonardo Ribeiro';

-- drop procedure Resumo_Pagamentos;
```