



Unioeste - Universidade Estadual do Oeste do Paraná

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Ciência da Computação

Curso de Bacharelado em Ciência da Computação

Docente: Matheus Raffael Simon

Trabalho Final – Aula 29

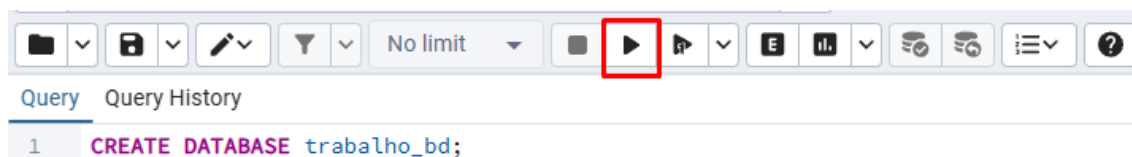
Pedro Henrique de Oliveira Berti

CASCADEL

2026

O trabalho foi feito usando o PostgreSQL como SGBD e o pgAdmin 4 como ferramenta gráfica. Usei a Query Tool em vez da PSQL, pois ela permite a execução das consultas de forma direta através da interface, sem a necessidade de digitar comandos completos no terminal.

Com isso, as consultas foram executadas apenas escrevendo o código na Query e clicando em executar.



Por esse motivo, os prints de execução mostram as consultas SQL e os resultados exibidos pela própria interface gráfica do pgAdmin, não é apresentado um comando explícito de execução, como ocorre no psql, uma vez que a ferramenta internamente envia as instruções ao banco de dados, sendo necessário apenas selecionar o código e executar.

Além do arquivo principal em formato .docx e do backup do banco de dados, estou enviando também a pasta “**Arquivos Extras**”, a qual contém todos os scripts SQL e as evidências de execução organizadas separadamente.

Essa pasta foi utilizada durante todo o desenvolvimento do trabalho para me organizar, servindo como repositório dos arquivos de criação da estrutura do banco, das consultas de cada questão e dos respectivos prints de execução. Sua inclusão na entrega tem como objetivo facilitar a conferência caso achar necessário e demonstrar o processo de organização ao longo do trabalho.

Antes de iniciar as questões, eu criei a estrutura do banco de dados com base no DER fornecido (AULA 28.png). Após a criação das tabelas, inseri dados fictícios por meio de comandos INSERT, com o objetivo de permitir a execução das consultas e ver os resultados.

A estrutura que criei está disponível em Arquivos Extras/SQL/estrutura.sql

Os inserts dos dados fictícios está disponível em Arquivos Extras/SQL/insert.sql

QUESTÃO 1

SELECT

u.nome AS nome_usuario,

u.email,

e.logradouro,

e.numero,

e.cep,

e.complemento,

t.numero AS telefone

FROM usuario u

LEFT JOIN endereco e ON e.idusuario = u.idusuario

LEFT JOIN telefone t ON t.idusuario = u.idusuario

ORDER BY u.idusuario;

Query

Query History

1

2

3

4

5

6

7

8

9

10

11

12

13

```
SELECT
  u.nome AS nome_usuario,
  u.email,
  e.logradouro,
  e.numero,
  e.cep,
  e.complemento,
  t.numero AS telefone
FROM usuario u
LEFT JOIN endereco e ON e.idusuario = u.idusuario
LEFT JOIN telefone t ON t.idusuario = u.idusuario
ORDER BY u.idusuario;
```

Scratch Pad

Data Output

Messages

Notifications

QUESTÃO 2

SELECT

```
p.nomefantasia,  
c.nome AS cidade,  
e.nome AS estado,  
e.sigla AS sigla_estado,  
pa.nome AS pais,  
pa.sigla AS sigla_pais
```

FROM parceiro p

JOIN endereco en ON en.idparceiro = p.idparceiro

JOIN cidade c ON c.idcidade = en.idcidade

JOIN estado e ON e.idestado = c.idestado

JOIN pais pa ON pa.idpais = e.idpais

ORDER BY p.idparceiro;

Query

Query History

Scratch Pad

1

2

3

4

5

6

7

8

9

10

11

12

13

14

```
SELECT
  p.nomefantasia,
  c.nome AS cidade,
  e.nome AS estado,
  e.sigla AS sigla_estado,
  pa.nome AS pais,
  pa.sigla AS sigla_pais
FROM parceiro p
JOIN endereco en ON en.idparceiro = p.idparceiro
JOIN cidade c ON c.idcidade = en.idcidade
JOIN estado e ON e.idestado = c.idestado
JOIN pais pa ON pa.idpais = e.idpais
ORDER BY p.idparceiro;
```

Data Output

Messages

Notifications

Showing rows: 1 to 3

Page No: 1 of 1

	nomefantasia character varying (100)	cidade character varying (100)	estado character varying (100)	sigla_estado character varying (10)	pais character varying (100)	sigla_pais character varying (10)
1	Los Pollos Hermanos	Cascavel	Paraná	PR	Brasil	BR
2	A1A Car Wash	Cascavel	Paraná	PR	Brasil	BR
3	Vamonos Pest	Cascavel	Paraná	PR	Brasil	BR

QUESTÃO 3

Foi utilizado peso X = 1.0 kg

Parceiro X = id 2000 (Los Pollos Hermanos)

SELECT DISTINCT

tp.idtabelapreco,

tp.nome AS nome_tabela_preco

FROM tabelapreco tp

JOIN tabelaprecoproduto tpp ON tpp.idtabelapreco = tp.idtabelapreco

JOIN produto p ON p.idproduto = tpp.idproduto

JOIN pedido ped ON ped.idtabelapreco = tp.idtabelapreco

WHERE

tp.idnativo = 0

AND p.peso > 1.0

AND ped.idparceiro = 2000;

The screenshot shows a SQL IDE interface. The top pane displays a SQL query with line numbers 1 through 12. The query is a SELECT DISTINCT statement that joins several tables: tabelapreco (tp), tabelaprecoproduto (tpp), produto (p), and pedido (ped). It filters for records where tp.idnativo is 0, p.peso is greater than 1.0, and ped.idparceiro is 2000. The bottom pane shows the 'Data Output' tab with a single row of results. The first column is 'idtabelapreco' (integer, primary key) with a value of 6000. The second column is 'nome_tabela_preco' (character varying, 100) with the value 'Tabela Sul - Ativa'.

```
1 SELECT DISTINCT
2     tp.idtabelapreco,
3     tp.nome AS nome_tabela_preco
4 FROM tabelapreco tp
5 JOIN tabelaprecoproduto tpp ON tpp.idtabelapreco = tp.idtabelapreco
6 JOIN produto p ON p.idproduto = tpp.idproduto
7 JOIN pedido ped ON ped.idtabelapreco = tp.idtabelapreco
8 WHERE
9     tp.idnativo = 0
10    AND p.peso > 1.0
11    AND ped.idparceiro = 2000;
12
```

idtabelapreco	nome_tabela_preco
6000	Tabela Sul - Ativa

QUESTÃO 4

SELECT

ped.idpedido AS numero_pedido,
ped.datapedido,
ped.valortotal,
ped.quantidadeprodutos,

par.nome AS nome_parceiro,
par.nomefantasia,

u.nome AS nome_usuario,
u.email,

tp.nome AS tabela_preco,
cp.nome AS condicao_pagamento,

prod.nome AS nome_produto,
prod.codigo AS codigo_produto,
pp.valorvenda,

pa.nome AS pais,
est.sigla AS sigla_estado,

tel.numero AS telefone_parceiro

FROM pedido ped

JOIN parceiro par ON par.idparceiro = ped.idparceiro

JOIN usuario u ON u.idusuario = ped.idusuario

JOIN tabelapreco tp ON tp.idtabelapreco = ped.idtabelapreco

JOIN condicaopagamento cp ON cp.idcondicaopagamento =
ped.idcondicaopagamento

JOIN pedidoproduto pp ON pp.idpedido = ped.idpedido

JOIN produto prod ON prod.idproduto = pp.idproduto

JOIN endereco endp ON endp.idparceiro = par.idparceiro

JOIN cidade cid ON cid.idcidade = endp.idcidade

JOIN estado est ON est.idestado = cid.idestado

JOIN pais pa ON pa.idpais = est.idpais

LEFT JOIN telefone tel ON tel.idparceiro = par.idparceiro

ORDER BY ped.idpedido;

Query

Query History

Scratch Pad

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

SELECT

ped.idpedido AS numero_pedido,

ped.datapedido,

ped.valortotal,

ped.quantidadeprodutos,

par.nome AS nome_parceiro,

par.nomefantasia,

u.nome AS nome_usuario,

u.email,

tp.nome AS tabela_preco,

cp.nome AS condicao_pagamento,

prod.nome AS nome_produto,

prod.codigo AS codigo_produto,

pp.valorvenda,

pa.nome AS pais,

est.sigla AS sigla_estado,

Data Output

Messages

Notifications

Showing rows: 1 to 6

Page No: 1 of 1

	numero_pedido integer	datapedido date	valortotal numeric (10,2)	quantidadeprodutos integer	nome_parceiro character varying (100)	nomefantasia character varying (100)	nome_usuario character varying (100)	email character varying (100)	tabela_preco character varying (100)
1	9000	2026-01-21	319.40	2	Los Pollos Hermanos Alimentos LTDA	Los Pollos Hermanos	Walter White	walterwhite@gmail.com	Tabela Sul - Ativa
2	9000	2026-01-21	319.40	2	Los Pollos Hermanos Alimentos LTDA	Los Pollos Hermanos	Walter White	walterwhite@gmail.com	Tabela Sul - Ativa
3	9001	2026-01-20	578.90	3	A1A Lavagem e Estetica Automotiva LT...	A1A Car Wash	Jesse Pinkman	jessepinkman@gmail.c...	Tabela Sul - Ativa
4	9001	2026-01-20	578.90	3	A1A Lavagem e Estetica Automotiva LT...	A1A Car Wash	Jesse Pinkman	jessepinkman@gmail.c...	Tabela Sul - Ativa
5	9001	2026-01-20	578.90	3	A1A Lavagem e Estetica Automotiva LT...	A1A Car Wash	Jesse Pinkman	jessepinkman@gmail.c...	Tabela Sul - Ativa
6	9002	2026-01-21	119.00	1	Vamonos Pest Controle Ambiental LTDA	Vamonos Pest	Saul Goodman	saugoodman@gmail.co...	Tabela Sul - Ativa

Query

Query History

Scratch Pad

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

SELECT

ped.idpedido AS numero_pedido,

ped.datapedido,

ped.valortotal,

ped.quantidadeprodutos,

par.nome AS nome_parceiro,

par.nomefantasia,

u.nome AS nome_usuario,

u.email,

tp.nome AS tabela_preco,

cp.nome AS condicao_pagamento,

prod.nome AS nome_produto,

prod.codigo AS codigo_produto,

pp.valorvenda,

pa.nome AS pais,

est.sigla AS sigla_estado,

Data Output

Messages

Notifications

Showing rows: 1 to 6

Page No: 1 of 1

	tabela_preco character varying (100)	condicao_pagamento character varying (100)	nome_produto character varying (100)	codigo_produto character varying (20)	valorvenda numeric (10,2)	pais character varying (100)	sigla_estado character varying (10)	telefone_parceiro character varying (20)
1	Tabela Sul - Ativa	PIX (mediato)	Máscara Respiratória com Filtro	AZ-HEIS-03	79.90	Brasil	PR	45 3225-9011
2	Tabela Sul - Ativa	PIX (mediato)	Kit Vidraria 500ml (Heisenberg L...	AZ-HEIS-01	189.90	Brasil	PR	45 3225-9011
3	Tabela Sul - Ativa	Cartão 2x	Máscara Respiratória com Filtro	AZ-HEIS-03	79.00	Brasil	PR	45 3035-7744
4	Tabela Sul - Ativa	Cartão 2x	Luvas Nitrílicas Caixa 100un	AZ-HEIS-04	54.90	Brasil	PR	45 3035-7744
5	Tabela Sul - Ativa	Cartão 2x	Carrinho Dobrável de Carga	AZ-HEIS-06	499.00	Brasil	PR	45 3035-7744
6	Tabela Sul - Ativa	Boleto 14 dias	Galão Plástico Reforçado 20L	AZ-HEIS-05	119.00	Brasil	PR	45 99973-5502

Query

Query History

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

```
    ped.ualaparcid,
    ped.valortotal,
    ped.quantidadeprodutos,

    par.nome AS nome_parceiro,
    par.nomefantasia,

    u.nome AS nome_usuario,
    u.email,

    tp.nome AS tabela_preco,
    cp.nome AS condicao_pagamento,

    prod.nome AS nome_produto,
    prod.codigo AS codigo_produto,
    pp.valorvenda,

    pa.nome AS pais,
    est.sigla AS sigla_estado,

    tel.numero AS telefone_parceiro

FROM pedido ped

JOIN parceiro par ON par.idparceiro = ped.idparceiro
JOIN usuario u ON u.idusuario = ped.idusuario
JOIN tabelapreco tp ON tp.idtabelapreco = ped.idtabelapreco
JOIN condicaopagamento cp ON cp.idcondicaopagamento = ped.idcondicaopagamento

JOIN pedidoproduto pp ON pp.idpedido = ped.idpedido
JOIN produto prod ON prod.idproduto = pp.idproduto

JOIN endereco endp ON endp.idparceiro = par.idparceiro
JOIN cidade cid ON cid.idcidade = endp.idcidade
JOIN estado est ON est.idestado = cid.idestado
JOIN pais pa ON pa.idpais = est.idpais

LEFT JOIN telefone tel ON tel.idparceiro = par.idparceiro

ORDER BY ped.idpedido;
```

Total rows: 6

Query complete 00:00:00.105

CRLF

Ln 15, Col 1

QUESTÃO 5

Tabela Transportadora:

```
CREATE TABLE transportadora (  
    idtransportadora SERIAL PRIMARY KEY,  
    nome VARCHAR(150),  
    cnpj VARCHAR(25)  
);
```

Tabela notafiscal:

```
CREATE TABLE notafiscal (  
    idnotafiscal SERIAL PRIMARY KEY,  
    numero_nf VARCHAR(20),  
    chave_acesso VARCHAR(60),  
    data_emissao DATE,  
    valor_total NUMERIC(10,2),  
    valor_icms NUMERIC(10,2),  
    base_icms NUMERIC(10,2),  
    valor_pis NUMERIC(10,2),  
    valor_cofins NUMERIC(10,2),  
  
    idpedido INTEGER REFERENCES pedido(idpedido),  
    idparceiro INTEGER REFERENCES parceiro(idparceiro),  
    idendereco INTEGER REFERENCES endereco(idendereco),  
    idtransportadora INTEGER REFERENCES transportadora(idtransportadora)  
);
```

Inserts Transportadora:

```
INSERT INTO transportadora (idtransportadora, nome, cnpj) VALUES  
(1, 'TransSul Logística LTDA', '12.345.678/0001-90'),  
(2, 'Rápido Oeste Transportes', '45.901.234/0001-55'),  
(3, 'Catarina Express', '98.765.432/0001-11'),  
(4, 'PR Cargo Distribuição', '11.222.333/0001-44'),  
(5, 'Fronteira Transportes', '66.777.888/0001-99');
```

Inserts notafiscal:

```
INSERT INTO notafiscal  
(numero_nf, chave_acesso, data_emissao, valor_total, valor_icms, base_icms,  
valor_pis, valor_cofins,  
idpedido, idparceiro, idendereco, idtransportadora)  
VALUES  
(  
'NF-84521',  
'41260193847561000190230000084521123456789012',  
CURRENT_DATE - 12, 319.40, 57.49, 319.40, 5.21, 24.11,  
9000, 2000, 3100, 1),  
  
(  
'NF-90377',  
'41260127483956000190230000090377456781230987',  
CURRENT_DATE - 11, 578.90, 104.20, 578.90, 9.11, 43.52,  
9001, 2001, 3101, 2),  
  
(  
'NF-11902',  
'41260156473829000190230000011902498372645123',  
CURRENT_DATE - 10, 119.00, 21.42, 119.00, 2.14, 8.93,  
9002, 2002, 3102, 3),  
  
(  
'NF-67218',
```

'41260193847561000190230000067218987654321098',
CURRENT_DATE - 9, 189.90, 34.18, 189.90, 3.42, 14.24,
9000, 2000, 3100, 4),

('NF-78455',
'41260101928374000190230000078455918273645109',
CURRENT_DATE - 8, 499.00, 89.82, 499.00, 7.48, 37.42,
9001, 2001, 3101, 5),

('NF-55091',
'41260191827364000190230000055091283746519082',
CURRENT_DATE - 7, 54.90, 9.88, 54.90, 0.82, 4.12,
9001, 2001, 3101, 1),

('NF-33807',
'41260110293847000190230000033807192837465019',
CURRENT_DATE - 6, 79.90, 14.38, 79.90, 1.20, 5.99,
9000, 2000, 3100, 2),

('NF-44629',
'41260156473829000190230000044629109283746510',
CURRENT_DATE - 5, 129.50, 23.31, 129.50, 1.94, 9.71,
9000, 2000, 3100, 3),

('NF-99102',
'41260193847561000190230000099102837465019283',
CURRENT_DATE - 4, 200.00, 36.00, 200.00, 3.00, 15.00,
9002, 2002, 3102, 4),

('NF-70544',
'41260101928374000190230000070544192837465098',
CURRENT_DATE - 3, 399.00, 71.82, 399.00, 5.98, 29.92,

9001, 2001, 3101, 5),

('NF-28419',

'41260191827364000190230000028419283746510982',

CURRENT_DATE - 2, 249.90, 44.98, 249.90, 3.74, 18.74,

9002, 2002, 3102, 1),

('NF-61073',

'41260110293847000190230000061073192837465012',

CURRENT_DATE - 2, 89.90, 16.18, 89.90, 1.35, 6.74,

9000, 2000, 3100, 2),

('NF-41786',

'41260156473829000190230000041786109283746598',

CURRENT_DATE - 1, 159.90, 28.78, 159.90, 2.40, 11.99,

9001, 2001, 3101, 3),

('NF-52890',

'41260193847561000190230000052890192837465087',

CURRENT_DATE - 1, 319.40, 57.49, 319.40, 5.21, 24.11,

9000, 2000, 3100, 4),

('NF-76014',

'41260101928374000190230000076014192837465031',

CURRENT_DATE, 578.90, 104.20, 578.90, 9.11, 43.52,

9001, 2001, 3101, 5);

```

1 CREATE TABLE transportadora (
2     idtransportadora SERIAL PRIMARY KEY,
3     nome VARCHAR(150),
4     cnpj VARCHAR(25)
5 );
6
7 CREATE TABLE notafiscal (
8     idnotafiscal SERIAL PRIMARY KEY,
9     numero_nf VARCHAR(20),
10    chave_acesso VARCHAR(60),
11    data_emissao DATE,
12    valor_total NUMERIC(10,2),
13    valor_icms NUMERIC(10,2),
14    base_icms NUMERIC(10,2),
15    valor_pis NUMERIC(10,2),
16    valor_cofins NUMERIC(10,2),
17
18    idpedido INTEGER REFERENCES pedido(idpedido),
19    idparceiro INTEGER REFERENCES parceiro(idparceiro),
20    idendereco INTEGER REFERENCES endereco(idendereco),
21    idtransportadora INTEGER REFERENCES transportadora(idtransportadora)
22 );
23

```

Total rows:	Query complete 00:00:00.109	CRLF	Ln 23, Col 1
-------------	-----------------------------	------	--------------

```
1 INSERT INTO transportadora (idtransportadora, nome, cnpj) VALUES
2 (1, 'TransSul Logística LTDA', '12.345.678/0001-90'),
3 (2, 'Rápido Oeste Transportes', '45.901.234/0001-55'),
4 (3, 'Catarina Express', '98.765.432/0001-11'),
5 (4, 'PR Cargo Distribuição', '11.222.333/0001-44'),
6 (5, 'Fronteira Transportes', '66.777.888/0001-99');
7
8 INSERT INTO notafiscal
9 (numero_nf, chave_acesso, data_emissao, valor_total, valor_icms, base_icms, valor_pis, valor_cofins,
10 idpedido, idparceiro, idendereco, idtransportadora)
11 VALUES
12 ('NF-84521',
13 '41260193847561000190230000084521123456789012',
14 CURRENT_DATE - 12, 319.40, 57.49, 319.40, 5.21, 24.11,
15 9000, 2000, 3100, 1),
16
17 ('NF-90377',
18 '41260127483956000190230000090377456781238987',
19 CURRENT_DATE - 11, 578.90, 104.20, 578.90, 9.11, 43.52,
20 9001, 2001, 3101, 2),
21
22 ('NF-11902',
23 '41260156473829000190230000011902408372645123',
24 CURRENT_DATE - 10, 119.00, 21.42, 119.00, 2.14, 8.93,
25 9002, 2002, 3102, 3),
```

Total rows: Query complete 00:00:00.065 CRLF Ln 86, Col 1

CRLF Ln 86, Col 1

Print Select:

Query

Query History

Scratch Pad

1

SELECT * FROM notafiscal;

2

SELECT * FROM transportadora;

3

Data Output

Messages

Notifications

Showing rows: 1 to 5

Page No: 1 of 1

	idtransportadora [PK] integer	nome character varying (150)	cnnpj character varying (25)
1	1	TransSul Logística LTDA	12.345.678/0001-90
2	2	Rápido Oeste Transport...	45.901.234/0001-55
3	3	Catarina Express	98.765.432/0001-11
4	4	PR Cargo Distribuição	11.222.333/0001-44
5	5	Fronteira Transportes	66.777.888/0001-99

✓

Successfully run. Total query runtime: 86 msec. 5 rows affected.

✕

Total rows: 5

Query complete 00:00:00.086

CRLF

Ln 3, Col 1

QUESTÃO 6

```
CREATE VIEW vw_relatorio_produto AS
SELECT
    prod.nome      AS nome_produto,
    par.documento  AS documento_parceiro,
    est.sigla      AS sigla_estado,
    tp.idtabelapreco AS codigo_tabela_preco,
    endu.cep       AS cep_usuario,
    tpp.preco      AS preco_produto,
    u.email        AS email_usuario
FROM pedido ped
JOIN usuario u ON u.idusuario = ped.idusuario
JOIN parceiro par ON par.idparceiro = ped.idparceiro
JOIN endereco endu ON endu.idusuario = u.idusuario
JOIN cidade cid ON cid.idcidade = endu.idcidade
JOIN estado est ON est.idestado = cid.idestado
JOIN tabelapreco tp ON tp.idtabelapreco = ped.idtabelapreco
JOIN tabelaprecoproduto tpp ON tpp.idtabelapreco = tp.idtabelapreco
JOIN produto prod ON prod.idproduto = tpp.idproduto;
```

Para testar a view:

```
SELECT * FROM vw_relatorio_produto
```


Print criar a View:

Query

Query History

Scratch Pad

X

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

CREATE VIEW vw_relatorio_produto AS

SELECT

prod.nome AS nome_produto,

par.documento AS documento_parceiro,

est.sigla AS sigla_estado,

tp.idtabelapreco AS codigo_tabela_preco,

endu.cep AS cep_usuario,

tpp.preco AS preco_produto,

u.email AS email_usuario

FROM pedido ped

JOIN usuario u ON u.idusuario = ped.idusuario

JOIN parceiro par ON par.idparceiro = ped.idparceiro

JOIN endereco endu ON endu.idusuario = u.idusuario

JOIN cidade cid ON cid.idcidade = endu.idcidade

JOIN estado est ON est.idestado = cid.idestado

JOIN tabelapreco tp ON tp.idtabelapreco = ped.idtabelapreco

JOIN tabelaprecoproduto tpp ON tpp.idtabelapreco = tp.idtabelapreco

JOIN produto prod ON prod.idproduto = tpp.idproduto;

Data Output

Messages

Notifications

CREATE VIEW

Query returned successfully in 88 msec.

Total rows:

Query complete 00:00:00.088

✓ Query returned successfully in 88 msec. X

CRLF

Ln 19, Col 1

Print testar a View:

trabalho_bd/postgres@PostgreSQL 18

No limit

Scratch Pad

X

1

2

SELECT * FROM vw_relatorio_produto;

Data Output

Messages

Notifications

Showing rows: 1 to 18

Page No: 1

of 1

	nome_produto character varying (100)	documento_parceiro character varying (20)	sigla_estado character varying (10)	codigo_tabela_preco integer	cep_usuario character varying (20)	preco_produto numeric (10,2)	email_usuario character varying (100)
1	Kit Vidraria 500ml (Heisenberg L...	10.684.395/0001-70	PR	6000	85814-120	189.90	saugoodman@gmail.co...
2	Kit Vidraria 500ml (Heisenberg L...	41.930.772/0001-08	PR	6000	85812-000	189.90	jessepinkman@gmail.c...
3	Kit Vidraria 500ml (Heisenberg L...	27.418.506/0001-93	PR	6000	85810-010	189.90	walterwhite@gmail.com
4	Balança Digital de Precisão 0.01g	10.684.395/0001-70	PR	6000	85814-120	129.50	saugoodman@gmail.co...
5	Balança Digital de Precisão 0.01g	41.930.772/0001-08	PR	6000	85812-000	129.50	jessepinkman@gmail.c...
6	Balança Digital de Precisão 0.01g	27.418.506/0001-93	PR	6000	85810-010	129.50	walterwhite@gmail.com
7	Máscara Respiratória com Filtro	10.684.395/0001-70	PR	6000	85814-120	79.90	saugoodman@gmail.co...
8	Máscara Respiratória com Filtro	41.930.772/0001-08	PR	6000	85812-000	79.90	jessepinkman@gmail.c...

Total rows: 18 Query complete 00:00:00.078 CRLF Ln 2, Col 1

	<div>nome_produto</div> <div>character varying (100)</div>	<div>documento_parceiro</div> <div>character varying (20)</div>	<div>sigla_estado</div> <div>character varying (10)</div>	<div>codigo_tabela_preco</div> <div>integer</div>	<div>cep_usuario</div> <div>character varying (20)</div>	<div>preco_produto</div> <div>numeric (10,2)</div>	<div>email_usuario</div> <div>character varying (100)</div>
1	Kit Vidraria 500ml (Heisenberg L...	10.684.395/0001-70	PR	6000	85814-120	189.90	saugoodman@gmail.co...
2	Kit Vidraria 500ml (Heisenberg L...	41.930.772/0001-08	PR	6000	85812-000	189.90	Jessepinkman@gmail.c...
3	Kit Vidraria 500ml (Heisenberg L...	27.418.506/0001-93	PR	6000	85810-010	189.90	walterwhite@gmail.com
4	Balança Digital de Precisão 0.01g	10.684.395/0001-70	PR	6000	85814-120	129.50	saugoodman@gmail.co...
5	Balança Digital de Precisão 0.01g	41.930.772/0001-08	PR	6000	85812-000	129.50	Jessepinkman@gmail.c...
6	Balança Digital de Precisão 0.01g	27.418.506/0001-93	PR	6000	85810-010	129.50	walterwhite@gmail.com
7	Máscara Respiratória com Filtro	10.684.395/0001-70	PR	6000	85814-120	79.90	saugoodman@gmail.co...
8	Máscara Respiratória com Filtro	41.930.772/0001-08	PR	6000	85812-000	79.90	Jessepinkman@gmail.c...
9	Máscara Respiratória com Filtro	27.418.506/0001-93	PR	6000	85810-010	79.90	walterwhite@gmail.com
10	Luvas Nitrílicas Caixa 100un	10.684.395/0001-70	PR	6000	85814-120	54.90	saugoodman@gmail.co...
11	Luvas Nitrílicas Caixa 100un	41.930.772/0001-08	PR	6000	85812-000	54.90	Jessepinkman@gmail.c...
12	Luvas Nitrílicas Caixa 100un	27.418.506/0001-93	PR	6000	85810-010	54.90	walterwhite@gmail.com
13	Gaílo Plástico Reforçado 20L	10.684.395/0001-70	PR	6000	85814-120	119.00	saugoodman@gmail.co...
14	Gaílo Plástico Reforçado 20L	41.930.772/0001-08	PR	6000	85812-000	119.00	Jessepinkman@gmail.c...
15	Gaílo Plástico Reforçado 20L	27.418.506/0001-93	PR	6000	85810-010	119.00	walterwhite@gmail.com
16	Carrinho Dobrável de Carga	10.684.395/0001-70	PR	6000	85814-120	499.00	saugoodman@gmail.co...
17	Carrinho Dobrável de Carga	41.930.772/0001-08	PR	6000	85812-000	499.00	Jessepinkman@gmail.c...
18	Carrinho Dobrável de Carga	27.418.506/0001-93	PR	6000	85810-010	499.00	walterwhite@gmail.com

QUESTÃO 7

```
CREATE OR REPLACE FUNCTION calcular_imc(  
    p_peso NUMERIC,  
    p_altura NUMERIC  
)  
RETURNS NUMERIC AS $$  
DECLARE  
    v_imc NUMERIC;  
BEGIN  
    v_imc := p_peso / (p_altura * p_altura);  
    RETURN ROUND(v_imc, 2);  
END;  
$$ LANGUAGE plpgsql;
```

Testar a Stored Procedure:

```
SELECT calcular_imc(80,1.80)
```


QUESTÃO 8

Stored Procedure para atualizar nome do produto pelo código:

```
CREATE OR REPLACE PROCEDURE sp_atualizar_nome_produto(  
    p_codigo VARCHAR,  
    p_novo_nome VARCHAR  
)  
LANGUAGE plpgsql  
AS $$  
BEGIN  
    UPDATE produto  
    SET nome = p_novo_nome  
    WHERE codigo = p_codigo;  
  
    IF NOT FOUND THEN  
        RAISE EXCEPTION 'Nenhum produto encontrado com o código: %', p_codigo;  
    END IF;  
END;  
$;
```

Testar:

```
CALL sp_atualizar_nome_produto('AZ-HEIS-02', 'Balança Digital de Precisão 0.01g  
(Rev A)');
```

Confirmar:

```
SELECT idproduto, codigo, nome
FROM produto
WHERE codigo = 'AZ-HEIS-02';
```

Stored Procedure para deletar produto pelo código:

```
CREATE OR REPLACE PROCEDURE sp_deletar_produto_por_codigo(
    p_codigo VARCHAR
)
LANGUAGE plpgsql
AS $$
DECLARE
    v_idproduto INTEGER;
BEGIN
    SELECT idproduto
    INTO v_idproduto
    FROM produto
    WHERE codigo = p_codigo;

    IF v_idproduto IS NULL THEN
        RAISE EXCEPTION 'Nenhum produto encontrado com o código: %', p_codigo;
    END IF;

    DELETE FROM pedidoproduto
    WHERE idproduto = v_idproduto;

    DELETE FROM tabelaprecoproduto
    WHERE idproduto = v_idproduto;
```

```
DELETE FROM produto
WHERE idproduto = v_idproduto;
END;
$$;
```

Teste (criei um produto “descartável”, para inserir e deletar ele):

```
INSERT INTO produto (codigo, nome, peso)
VALUES ('DEL-TEST-45', 'Produto para teste de deleção', 0.99);
```

```
INSERT INTO tabelaprecoproduto (idtabelapreco, idproduto, preco)
SELECT 6000, idproduto, 9.90
FROM produto
WHERE codigo = 'DEL-TEST-45';
```

```
CALL sp_deletar_produto_por_codigo('DEL-TEST-45');
```

```
SELECT * FROM produto WHERE codigo = 'DEL-TEST-45';
```

Print Stored Procedure para atualizar nome do produto pelo código:

trabalho_bd/postgres@PostgreSQL 18

No limit

Query Query HistoryScratch Pad x

1CREATE OR REPLACE PROCEDURE sp_atualizar_nome_produto(
2 p_codigo VARCHAR,
3 p_novo_nome VARCHAR
4)
5LANGUAGE plpgsql
6AS \$\$
7BEGIN
8 UPDATE produto
9 SET nome = p_novo_nome
10 WHERE codigo = p_codigo;
11
12 IF NOT FOUND THEN
13 RAISE EXCEPTION 'Nenhum produto encontrado com o código: %', p_codigo;
14 END IF;
15END;
16\$\$;
17

Data OutputMessagesNotifications

CREATE PROCEDURE

Query returned successfully in 93 msec.

Total rows: Query complete 00:00:00.093

✓ Query returned successfully in 93 msec. ✕

CRLFLn 17, Col

Print Testar:

[illegible]

Print Confirmar:

[illegible]

Print Stored Procedure para deletar produto pelo código:

The screenshot shows a PostgreSQL IDE window titled "trabalho_bd/postgres@PostgreSQL 18". The main editor displays a SQL script for creating a stored procedure. The script is as follows:

```
1 /
2
3
4 LANGUAGE plpgsql
5 AS $$
6 DECLARE
7     v_idproduto INTEGER;
8 BEGIN
9     SELECT idproduto
10    INTO v_idproduto
11   FROM produto
12  WHERE codigo = p_codigo;
13
14 IF v_idproduto IS NULL THEN
15     RAISE EXCEPTION 'Nenhum produto encontrado com o código: %', p_codigo;
16 END IF;
17
18 | DELETE FROM pedidoproduto
19   WHERE idproduto = v_idproduto;
20
21 DELETE FROM tabelaprecoproduto
22   WHERE idproduto = v_idproduto;
23
24 DELETE FROM produto
25   WHERE idproduto = v_idproduto;
26 END;
27 $$;
28
```

Below the editor, the "Data Output" tab is active, showing the message: "CREATE PROCEDURE". Below that, a status message reads: "Query returned successfully in 78 msec."

At the bottom of the IDE, a status bar indicates: "Total rows: Query complete 00:00:00.078" on the left, and "CRLF Ln 18, Col 1" on the right.

Print produto “descartável”, para inserir e deletar ele:

trabalho_bd/postgres@PostgreSQL 18

Query HistoryScratch Pad X

```
1 INSERT INTO produto (codigo, nome, peso)
2 VALUES ('DEL-TEST-45', 'Produto para teste de deleção', 0.99);
3
4 INSERT INTO tabelaprecoproduto (idtabelapreco, idproduto, preco)
5 SELECT 6000, idproduto, 9.90
6 FROM produto
7 WHERE codigo = 'DEL-TEST-45';
8
9 CALL sp_deletar_produto_por_codigo('DEL-TEST-45');
10
11 SELECT * FROM produto WHERE codigo = 'DEL-TEST-45';
12
```

Data OutputMessagesNotifications

SQL

idproduto [PK] integer	codigo character varying (20)	nome character varying (100)	peso numeric (10,2)
---------------------------	----------------------------------	---------------------------------	------------------------

Total rows: 0 Query complete 00:00:00.076

✓ Successfully run. Total query runtime: 76 msec. 0 rows affected.

QUESTÃO 9

```
CREATE TABLE pedido_audit (  
    idpedido_audit SERIAL PRIMARY KEY,  
  
    -- campos atuais  
    idusuario_atual INTEGER,  
    idparceiro_atual INTEGER,  
    idtabelapreco_atual INTEGER,  
    idcondicaopagamento_atual INTEGER,  
    valortotal_atual NUMERIC(10,2),  
    quantidadeprodutos_atual INTEGER,  
  
    -- campos antigos  
    idusuario_old INTEGER,  
    idparceiro_old INTEGER,  
    idtabelapreco_old INTEGER,  
    idcondicaopagamento_old INTEGER,  
    valortotal_old NUMERIC(10,2),  
    quantidadeprodutos_old INTEGER,  
  
    ultima_atualizacao TIMESTAMP,  
    ultima_operacao VARCHAR(20)  
);
```

Trigger de auditoria (INSERT, UPDATE, DELETE):

```
CREATE OR REPLACE FUNCTION fn_auditoria_pedido()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO pedido_audit (
            idusuario_atual, idparceiro_atual, idtabelapreco_atual,
            idcondicaopagamento_atual,
            valortotal_atual, quantidadeprodutos_atual,
            ultima_atualizacao, ultima_operacao
        )
        VALUES (
            NEW.idusuario, NEW.idparceiro, NEW.idtabelapreco,
            NEW.idcondicaopagamento,
            NEW.valortotal, NEW.quantidadeprodutos,
            NOW(), 'Insert'
        );
        RETURN NEW;

    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO pedido_audit (
            idusuario_atual, idparceiro_atual, idtabelapreco_atual,
            idcondicaopagamento_atual,
            valortotal_atual, quantidadeprodutos_atual,
            idusuario_old, idparceiro_old, idtabelapreco_old, idcondicaopagamento_old,
            valortotal_old, quantidadeprodutos_old,
            ultima_atualizacao, ultima_operacao
        )
        VALUES (
            NEW.idusuario, NEW.idparceiro, NEW.idtabelapreco,
            NEW.idcondicaopagamento,
            NEW.valortotal, NEW.quantidadeprodutos,
```

```

        OLD.idusuario, OLD.idparceiro, OLD.idtabelapreco,
        OLD.idcondicaopagamento,
        OLD.valortotal, OLD.quantidadeprodutos,
        NOW(), 'Atualizado'
    );
    RETURN NEW;

ELSIF TG_OP = 'DELETE' THEN
    INSERT INTO pedido_audit (
        idusuario_old, idparceiro_old, idtabelapreco_old, idcondicaopagamento_old,
        valortotal_old, quantidadeprodutos_old,
        ultima_atualizacao, ultima_operacao
    )
    VALUES (
        OLD.idusuario, OLD.idparceiro, OLD.idtabelapreco,
        OLD.idcondicaopagamento,
        OLD.valortotal, OLD.quantidadeprodutos,
        NOW(), 'Deletado'
    );
    RETURN OLD;
END IF;
END;
$$ LANGUAGE plpgsql;

```

Criar a trigger:

```

CREATE TRIGGER trg_auditoria_pedido
AFTER INSERT OR UPDATE OR DELETE ON pedido
FOR EACH ROW
EXECUTE FUNCTION fn_auditoria_pedido();

```

Código para provar, (de forma automática), que a auditoria está funcionando corretamente para todas as operações possíveis:

```
DROP TABLE IF EXISTS _audit_test_marker;
```

```
CREATE TEMP TABLE _audit_test_marker (  
    started_at TIMESTAMP NOT NULL,  
    idpedido_criado INTEGER NOT NULL  
) ON COMMIT PRESERVE ROWS;
```

```
DO $$
```

```
DECLARE
```

```
    v_idpedido INTEGER;
```

```
BEGIN
```

```
    INSERT INTO pedido
```

```
        (datapedido, valortotal, quantidadeprodutos, idusuario, idparceiro, idtabelapreco,  
        idcondicaopagamento)
```

```
    VALUES
```

```
        (CURRENT_DATE, 149.90, 1, 1000, 2000, 6000, 7000)
```

```
    RETURNING idpedido INTO v_idpedido;
```

```
    INSERT INTO _audit_test_marker (started_at, idpedido_criado)
```

```
    VALUES (NOW(), v_idpedido);
```

```
    UPDATE pedido
```

```
    SET valortotal = valortotal + 20,
```

```
        quantidadeprodutos = quantidadeprodutos + 1
```

```
    WHERE idpedido = v_idpedido;
```

```
    INSERT INTO pedidoproduto (idpedido, idproduto, valorvenda)
```

```
    VALUES (v_idpedido, 5002, 79.90);
```

```

DELETE FROM pedidoproduto
WHERE idpedido = v_idpedido;

DELETE FROM pedido
WHERE idpedido = v_idpedido;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM _audit_test_marker;

SELECT
    idpedido_audit,
    valortotal_atual, valortotal_old,
    quantidadeprodutos_atual, quantidadeprodutos_old,
    ultima_atualizacao,
    ultima_operacao
FROM pedido_audit
WHERE ultima_atualizacao >= (SELECT started_at FROM _audit_test_marker LIMIT
1)
ORDER BY idpedido_audit;

```

Trigger de validação:

```

CREATE OR REPLACE FUNCTION fn_validar_pedido()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.valortotal < 1 THEN
        RAISE EXCEPTION 'Valor total não pode ser menor que 1';
    END IF;

    IF NEW.quantidadeprodutos < 0 THEN
        RAISE EXCEPTION 'Quantidade de produtos não pode ser negativa';
    END IF;

```



```
END IF;
```

```
IF NEW.datapedido < CURRENT_DATE THEN
```

```
    RAISE EXCEPTION 'Data do pedido não pode ser menor que a data atual';
```

```
END IF;
```

```
RETURN NEW;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

Criar trigger de validação:

```
CREATE TRIGGER trg_validar_pedido
```

```
BEFORE INSERT OR UPDATE ON pedido
```

```
FOR EACH ROW
```

```
EXECUTE FUNCTION fn_validar_pedido();
```

Testes de erro

Valor inválido:

```
INSERT INTO pedido
```

```
(datapedido, valortotal, quantidadeprodutos, idusuario, idparceiro, idtabelapreco,  
idcondicaopagamento)
```

```
VALUES
```

```
(CURRENT_DATE, 0.50, 1, 1000, 2000, 6000, 7000);
```

Quantidade inválida:

```
INSERT INTO pedido
```

```
(datapedido, valortotal, quantidadeprodutos, idusuario, idparceiro, idtabelapreco,  
idcondicaopagamento)
```

```
VALUES
```

```
(CURRENT_DATE, 100, -1, 1000, 2000, 6000, 7000);
```

Data inválida:

```
INSERT INTO pedido
```

```
(datapedido, valortotal, quantidadeprodutos, idusuario, idparceiro, idtabelapreco,  
idcondicaopagamento)
```

```
VALUES
```

```
(CURRENT_DATE - 5, 100, 1, 1000, 2000, 6000, 7000);
```

Print audit:

trabalho_bd/postgres@PostgreSQL 18

No limit

Query

Query History

Scratch Pad X

```
1 CREATE TABLE pedido_audit (  
2     idpedido_audit SERIAL PRIMARY KEY,  
3  
4     -- campos atuais  
5     idusuario_atual INTEGER,  
6     idparceiro_atual INTEGER,  
7     idtabelapreco_atual INTEGER,  
8     idcondicaopagamento_atual INTEGER,  
9     valortotal_atual NUMERIC(10,2),  
10    quantidadeprodutos_atual INTEGER,  
11  
12    -- campos antigos  
13    idusuario_old INTEGER,  
14    idparceiro_old INTEGER,  
15    idtabelapreco_old INTEGER,  
16    idcondicaopagamento_old INTEGER,  
17    valortotal_old NUMERIC(10,2),  
18    quantidadeprodutos_old INTEGER,  
19  
20    ultima_atualizacao TIMESTAMP,  
21    ultima_operacao VARCHAR(20)  
22 );  
23
```

Data Output

Messages

Notifications

CREATE TABLE

Query returned successfully in 94 msec.

Total rows: Query complete 00:00:00.094

✓ Query returned successfully in 94 msec. ✕

CRLF Ln 23, Col 1

Print Trigger de auditoria (INSERT, UPDATE, DELETE):

Dashboard x Properties x SQL x Statistics x Dependencies x Dependents x Processes x trabalho_bd/postgres@PostgreSQL 18* x

trabalho_bd/postgres@PostgreSQL 18

No limit

Query Query History

Scratch Pad x

```
1 CREATE OR REPLACE FUNCTION fn_auditoria_pedido()
2 RETURNS TRIGGER AS $$
3 BEGIN
4     IF TG_OP = 'INSERT' THEN
5         INSERT INTO pedido_audit (
6             idusuario_atual, idparceiro_atual, idtabelapreco_atual, idcondicaopagamento_atual,
7             valortotal_atual, quantidadeprodutos_atual,
8             ultima_atualizacao, ultima_operacao
9         )
10        VALUES (
11            NEW.idusuario, NEW.idparceiro, NEW.idtabelapreco, NEW.idcondicaopagamento,
12            NEW.valortotal, NEW.quantidadeprodutos,
13            NOW(), 'Insert'
14        );
15        RETURN NEW;
16
17     ELSIF TG_OP = 'UPDATE' THEN
18         INSERT INTO pedido_audit (
19             idusuario_atual, idparceiro_atual, idtabelapreco_atual, idcondicaopagamento_atual,
20             valortotal_atual, quantidadeprodutos_atual,
21             idusuario_old, idparceiro_old, idtabelapreco_old, idcondicaopagamento_old,
22             valortotal_old, quantidadeprodutos_old,
23             ultima_atualizacao, ultima_operacao
24         )
25        VALUES (
26            NEW.idusuario, NEW.idparceiro, NEW.idtabelapreco, NEW.idcondicaopagamento,
27            NEW.valortotal, NEW.quantidadeprodutos,
28            OLD.idusuario, OLD.idparceiro, OLD.idtabelapreco, OLD.idcondicaopagamento,
29            OLD.valortotal, OLD.quantidadeprodutos,
30            NOW(), 'Update'
31        );
32        RETURN NEW;
33
34     IF TG_OP = 'DELETE' THEN
35         INSERT INTO pedido_audit (
36             idusuario_atual, idparceiro_atual, idtabelapreco_atual, idcondicaopagamento_atual,
37             valortotal_atual, quantidadeprodutos_atual,
38             idusuario_old, idparceiro_old, idtabelapreco_old, idcondicaopagamento_old,
39             valortotal_old, quantidadeprodutos_old,
40             ultima_atualizacao, ultima_operacao
41         )
42        VALUES (
43            OLD.idusuario, OLD.idparceiro, OLD.idtabelapreco, OLD.idcondicaopagamento,
44            OLD.valortotal, OLD.quantidadeprodutos,
45            OLD.idusuario, OLD.idparceiro, OLD.idtabelapreco, OLD.idcondicaopagamento,
46            OLD.valortotal, OLD.quantidadeprodutos,
47            NOW(), 'Delete'
48        );
49        RETURN OLD;
50
51     RETURN NEW;
52 END;
```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 65 msec.

Total rows: Query complete 00:00:00.065

✓ Query returned successfully in 65 msec. ✕

CRLF Ln 49, Col 1

Print Criar a trigger:

QueryQuery History

1CREATE TRIGGER trg_auditoria_pedido

2AFTER INSERT OR UPDATE OR DELETE ON pedido

3FOR EACH ROW

4EXECUTE FUNCTION fn_auditoria_pedido();

5

Scratch Pad x

Data OutputMessagesNotifications

CREATE TRIGGER

Query returned successfully in 66 msec.

Total rows:Query complete 00:00:00.066

✓ Query returned successfully in 66 msec. ✕

CRLF Ln 5, Col 1

Print da auditoria funcionando corretamente para todas as operações possíveis:

trabalho_bd/postgres@PostgreSQL 18

Query History

```
25  
26 INSERT INTO pedidoproduto (idpedido, idproduto, valorvenda)  
27 VALUES (v_idpedido, 5002, 79.90);  
  
28  
29  
30 DELETE FROM pedidoproduto  
31 WHERE idpedido = v_idpedido;  
32  
33 DELETE FROM pedido  
34 WHERE idpedido = v_idpedido;  
35  
36 END;  
37 $$ LANGUAGE plpgsql;  
38  
39 SELECT * FROM _audit_test_marker;  
40  
41 SELECT  
42     idpedido_audit,  
43     valortotal_atual, valortotal_oldd,  
44     quantidadeprodutos_atual, quantidadeprodutos_old,  
45     ultima_atualizacao,  
46     ultima_operacao  
47 FROM pedido_audit  
48 WHERE ultima_atualizacao >= (SELECT started_at FROM _audit_test_marker LIMIT 1)  
49 ORDER BY idpedido_audit;
```

Scratch Pad

Data Output Messages Notifications

Showing rows: 1 to 3 Page No: 1 of 1

idpedido_audit [PK] integer	valortotal_atual numeric (10,2)	valortotal_oldd numeric (10,2)	quantidadeprodutos_atual integer	quantidadeprodutos_old integer	ultima_atualizacao timestamp without time zone	ultima_operacao character varying (20)
1	2	149.90	[null]	1	2026-01-23 02:12:28.681542	Insert
2	3	169.90	149.90	2	2026-01-23 02:12:28.681542	Atualizado
3	4	[null]	169.90	[null]	2026-01-23 02:12:28.681542	Deletado

Total rows: 3 Query complete 00:00:00.084

Successfully run. Total query runtime: 84 msec. 3 rows affected.

CRLF Ln 49, Col 1

Print Trigger de validação:

Dashboard × Properties × SQL × Statistics × Dependencies × Dependents × Processes × trabalho_bd/postgres@PostgreSQL 18*

trabalho_bd/postgres@PostgreSQL 18

No limit

Query Query History Scratch Pad

```

1 CREATE OR REPLACE FUNCTION fn_validar_pedido()
2 RETURNS TRIGGER AS $$
3 BEGIN
4     IF NEW.valortotal < 1 THEN
5         RAISE EXCEPTION 'Valor total não pode ser menor que 1';
6     END IF;
7
8     IF NEW.quantidadeprodutos < 0 THEN
9         RAISE EXCEPTION 'Quantidade de produtos não pode ser negativa';
10    END IF;
11
12   IF NEW.datapedido < CURRENT_DATE THEN
13       RAISE EXCEPTION 'Data do pedido não pode ser menor que a data atual';
14   END IF;
15
16   RETURN NEW;
17 END;
18 $$ LANGUAGE plpgsql;
19 
```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 84 msec.

Total rows: Query complete 00:00:00.084 CRLF Ln 19, Col 1

Print Criar trigger de validação:

Dashboard x Properties x SQL x Statistics x Dependencies x Dependents x Processes x trabalho_bd/postgres@PostgreSQL 18*

trabalho_bd/postgres@PostgreSQL 18

No limit

Query History

```

1 CREATE TRIGGER trg_validar_pedido
2 BEFORE INSERT OR UPDATE ON pedido
3 FOR EACH ROW
4 EXECUTE FUNCTION fn_validar_pedido();
5

```

Scratch Pad

Data Output Messages Notifications

CREATE TRIGGER

Query returned successfully in 64 msec.

Total rows: Query complete 00:00:00.064

✓ Query returned successfully in 64 msec.

CRLF Ln 5, Col

Print Testes de erro

Valor inválido:

[Query](#) [Query History](#)

```
1  INSERT INTO pedido
2  (datapedido, valortotal, quantidadeprodutos, idusuario, idparceiro, idtabelapreco, idcondicaopagamento)
3  VALUES
4  (CURRENT_DATE, 0.50, 1, 1000, 2000, 6000, 7000);
5
```

[Data Output](#) [Messages](#) [Notifications](#)

ERROR: Valor total não pode ser menor que 1
CONTEXT: PL/pgSQL function fn_validar_pedido() line 4 at RAISE

ERRO: Valor total não pode ser menor que 1
SQL state: P0001

Print Quantidade inválida:

Query

Query History

1

2

3

4

5

INSERT INTO pedido

(datapedido, valortotal, quantidadeprodutos, idusuario, idparceiro, idtabelapreco, idcondicaopagamento)

VALUES

(CURRENT_DATE, 100, -1, 1000, 2000, 6000, 7000);

Data Output

Messages

Notifications

ERROR: Quantidade de produtos não pode ser negativa

CONTEXT: PL/pgSQL function fn_validar_pedido() line 8 at RAISE

ERRO: Quantidade de produtos não pode ser negativa

SQL state: P0001

Print Data inválida:

Query

Query History

1

INSERT INTO pedido

2

(datapedido, valortotal, quantidadeprodutos, idusuario, idparceiro, idtabelapreco, idcondicaopagamento)

3

VALUES

4

(CURRENT_DATE - 5, 100, 1, 1000, 2000, 6000, 7000);

5

Data Output

Messages

Notifications

ERROR: Data do pedido não pode ser menor que a data atual

CONTEXT: PL/pgSQL function fn_validar_pedido() line 12 at RAISE

ERR0: Data do pedido não pode ser menor que a data atual

SQL state: P0001