



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

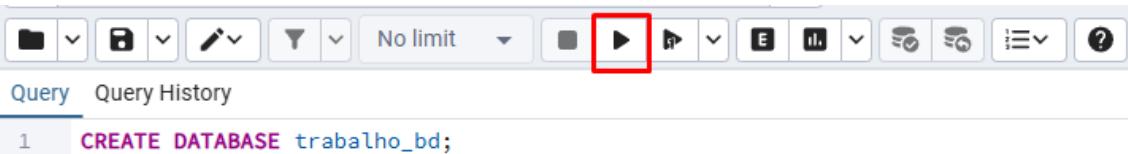
Pedro Henrique de Oliveira Berti

CASCABEL

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.



```
Query History
```

```
1 CREATE DATABASE trabalho_bd;
```

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.

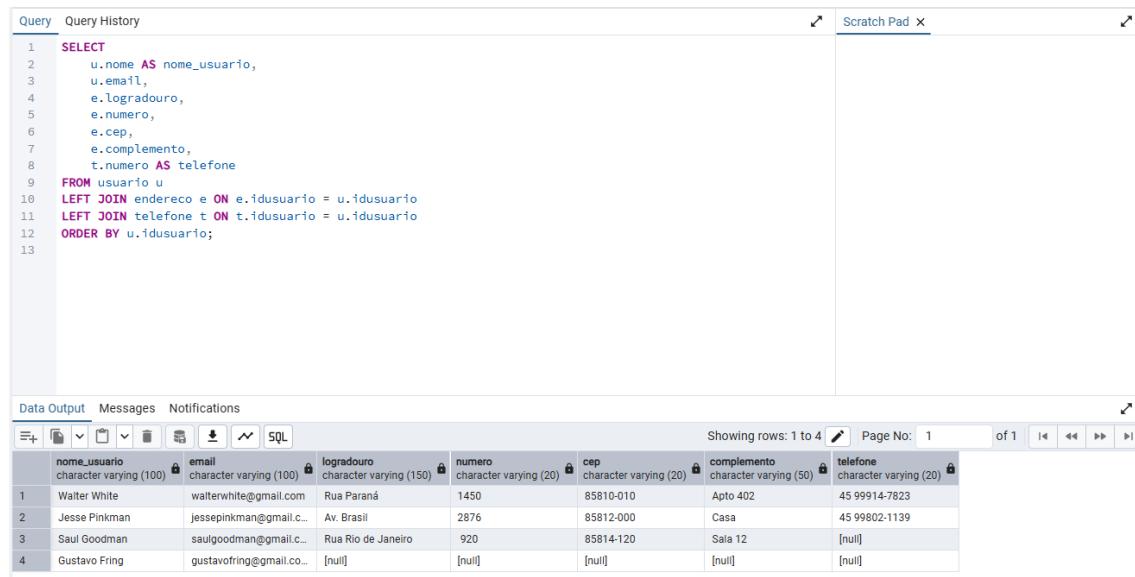
A estrutura que criei está disponível em Arquivos Extras/SQL/estrutura.sql

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;
```



The screenshot shows the MySQL Workbench interface with two tabs: 'Query' and 'Scratch Pad'. The 'Query' tab contains the SQL code provided above. The 'Data Output' tab displays the results of the query, which are as follows:

	nome_usuario	email	logradouro	numero	cep	complemento	telefone
1	Walter White	walterwhite@gmail.com	Rua Paraná	1450	85810-010	Apto 402	45 99914-7823
2	Jesse Pinkman	jessepinkman@gmail.c...	Av. Brasil	2876	85812-000	Casa	45 99802-1139
3	Saul Goodman	saulgoodman@gmail.c...	Rua Rio de Janeiro	920	85814-120	Sala 12	[null]
4	Gustavo Fring	gustavofring@gmail.co...	[null]	[null]	[null]	[null]	[null]

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 History

```
1  SELECT  
2      p.nomefantasia,  
3      c.nome AS cidade,  
4      e.nome AS estado,  
5      e.sigla AS sigla_estado,  
6      pa.nome AS pais,  
7      pa.sigla AS sigla_pais  
8  FROM parceiro p  
9  JOIN endereco en ON en.idparceiro = p.idparceiro  
10 JOIN cidade c ON c.idcidade = en.idcidade  
11 JOIN estado e ON e.idestado = c.idestado  
12 JOIN pais pa ON pa.idpais = e.idpais  
13 ORDER BY p.idparceiro;
```

Scratch Pad

Data Output

nomefantasia	cidade	estado	sigla_estado	pais	sigla_pais
Los Pollos Hermanos	Cascavel	Paraná	PR	Brasil	BR
A1A Car Wash	Cascavel	Paraná	PR	Brasil	BR
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 query editor interface with two tabs: 'Query' and 'Scratch Pad'. The 'Query' tab contains the SQL code provided above. The 'Scratch Pad' tab is empty. Below the tabs is a toolbar with icons for file operations and a 'SQL' button. The main area displays the results of the query in a table titled 'tabelapreco'. The table has two columns: 'idtabelapreco' (PK Integer) and 'nome_tabela_preco' (character varying (100)). A single row is shown with the value 6000 in the first column and 'Tabela Sul - Ativa' in the second column. At the bottom, there are buttons for 'Data Output', 'Messages', and 'Notifications', along with pagination controls showing 'Showing rows: 1 to 1' and 'Page No: 1 of 1'.

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  SELECT
2      ped.idpedido AS numero_pedido,
3      ped.datapedido,
4      ped.valortotal,
5      ped.quantidadeprodutos,
6
7      par.nome AS nome_parceiro,
8      par.nomefantasia,
9
10     u.nome AS nome_usuario,
11     u.email,
12
13     tp.nome AS tabela_preco,
14     cp.nome AS condicao_pagamento,
15
16     prod.nome AS nome_produto,
17     prod.codigo AS codigo_produto,
18     pp.valorvenda,
19
20     pa.nome AS pais,
21     est.sigla AS sigla_estado,

```

Data Output Messages Notifications

	numero_pedido	datapedido	valortotal	quantidadeprodutos	nome_parceiro	nomefantasia	nome_usuario	email	tabela_preco
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	saulgoodman@gmail.co...	Tabela Sul - Ativa

Query Query History Scratch Pad

```

1  SELECT
2      ped.idpedido AS numero_pedido,
3      ped.datapedido,
4      ped.valortotal,
5      ped.quantidadeprodutos,
6
7      par.nome AS nome_parceiro,
8      par.nomefantasia,
9
10     u.nome AS nome_usuario,
11     u.email,
12
13     tp.nome AS tabela_preco,
14     cp.nome AS condicao_pagamento,
15
16     prod.nome AS nome_produto,
17     prod.codigo AS codigo_produto,
18     pp.valorvenda,
19
20     pa.nome AS pais,
21     est.sigla AS sigla_estado,

```

Data Output Messages Notifications

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

Total rows: 6 Query complete 00:00:00.105 CRLF Ln 15, Col 1

```
Query History
3     ped.datapedido,
4     ped.valortotal,
5     ped.quantidadeprodutos,
6
7     par.nome AS nome_parceiro,
8     par.nomefantasia,
9
10    u.nome AS nome_usuario,
11    u.email,
12
13    tp.nome AS tabela_preco,
14    cp.nome AS condicao_pagamento,
15
16    prod.nome AS nome_produto,
17    prod.codigo AS codigo_produto,
18    pp.valorvenda,
19
20    pa.nome AS pais,
21    est.sigla AS sigla_estado,
22
23    tel.numero AS telefone_parceiro
24
25 FROM pedido ped
26
27 JOIN parceiro par ON par.idparceiro = ped.idparceiro
28 JOIN usuario u ON u.idusuario = ped.idusuario
29 JOIN tabelapreco tp ON tp.idtabelapreco = ped.idtabelapreco
30 JOIN condicaopagamento cp ON cp.idcondicaopagamento = ped.idcondicaopagamento
31
32 JOIN pedidoproduto pp ON pp.idpedido = ped.idpedido
33 JOIN produto prod ON prod.idproduto = pp.idproduto
34
35 JOIN endereco endp ON endp.idparceiro = par.idparceiro
36 JOIN cidade cid ON cid.idcidade = endp.idcidade
37 JOIN estado est ON est.idestado = cid.idestado
38 JOIN pais pa ON pa.idpais = est.idpais
39
40 LEFT JOIN telefone tel ON tel.idparceiro = par.idparceiro
41
42 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);

Print Tabelas:

The screenshot shows a PostgreSQL query editor interface. The left pane contains the SQL code for creating two tables: `transportadora` and `notafiscal`. The right pane is labeled "Scratch Pad". Below the editor, the status bar indicates "Data Output" is selected, showing the query was executed successfully in 109 msec. The bottom status bar shows "Total rows: 0" and "Query complete 00:00:00.109".

```
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
```

CREATE TABLE

Query returned successfully in 109 msec.

Total rows: 0 Query complete 00:00:00.109 CRLF Ln 23, Col 1

Print Inserts:

The screenshot shows a PostgreSQL query editor interface. The left pane contains the SQL code for inserting data into the `transportadora` and `notafiscal` tables. The right pane is labeled "Scratch Pad". Below the editor, the status bar indicates "Data Output" is selected, showing the query was executed successfully in 65 msec. The bottom status bar shows "Total rows: 0" and "Query complete 00:00:00.065".

```
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  '41266193847561000190230000084521123456789012',
14  CURRENT_DATE - 12, 319.40, 57.49, 319.40, 5.21, 24.11,
15  9000, 2000, 3100, 1),
16
17 ('NF-98377',
18  '41266127483956000190230000090377456781230987',
19  CURRENT_DATE - 11, 578.90, 104.20, 578.90, 9.11, 43.52,
20  9001, 2001, 3101, 2),
21
22 ('NF-11982',
23  '41266156473829000190230000011902498372645123',
24  CURRENT_DATE - 10, 119.00, 21.42, 119.00, 2.14, 8.93,
25  9002, 2002, 3102, 3),
```

INSERT 0 15

Query returned successfully in 65 msec.

Total rows: 0 Query complete 00:00:00.065 CRLF Ln 86, Col 1

Print Select:

Query 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 1 | 2 | 3 | 4 | 5 |

	idtransportadora [PK] integer	nome character varying (150)	cnpj character varying (25)
1	1	TransSul Logistica 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

Total rows: 5 Query complete 00:00:00.086 ✓ Successfully run. Total query runtime: 86 msec. 5 rows affected. CRLF Ln 3, Col 1

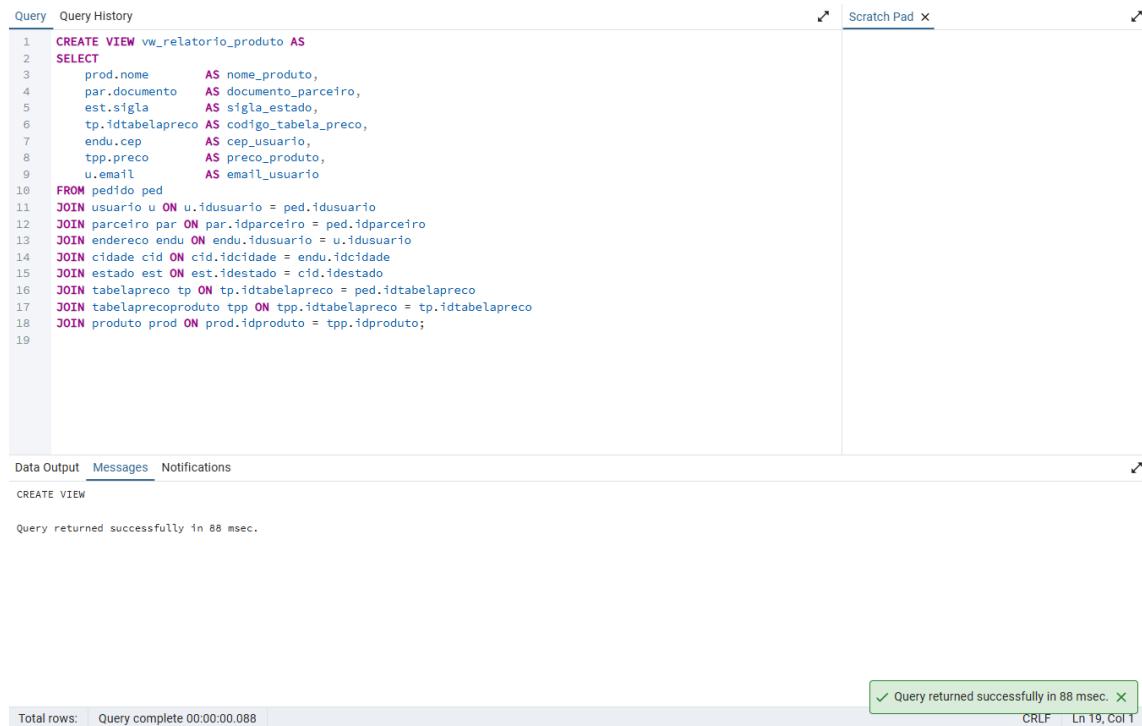
QUESTÃO 6

```
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:



```

1 CREATE VIEW vw_relatorio_produto AS
2 SELECT
3     prod.nome      AS nome_produto,
4     par.documento AS documento_parceiro,
5     est.sigla      AS sigla_estado,
6     tp.idtabelapreco AS codigo_tabela_preco,
7     endu.cep       AS cep_usuario,
8     tpp.preco     AS preco_produto,
9     u.email        AS email_usuario
10    FROM pedido ped
11    JOIN usuario u ON u.idusuario = ped.idusuario
12    JOIN parceiro par ON par.idparceiro = ped.idparceiro
13    JOIN enderecos endu ON endu.idusuario = u.idusuario
14    JOIN cidades cid ON cid.idcidade = endu.idcidade
15    JOIN estados est ON est.idestado = cid.idestado
16    JOIN tabelapreco tp ON tp.idtabelapreco = ped.idtabelapreco
17    JOIN tabelaprecoproduto tpp ON tpp.idtabelapreco = tp.idtabelapreco
18    JOIN produto prod ON prod.idproduto = tpp.idproduto;
19

```

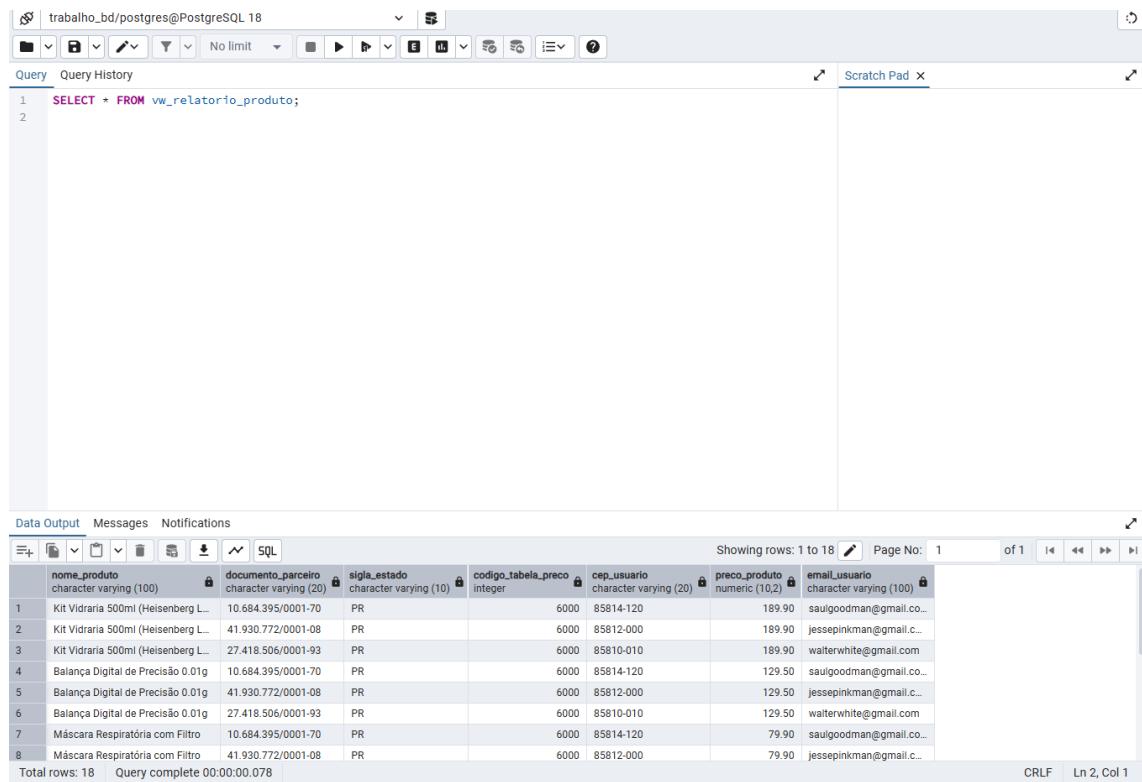
Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 88 msec.

Total rows: Query complete 00:00:00.088 ✓ CRLF Ln 19, Col 1

Print testar a View:



```

1 SELECT * FROM vw_relatorio_produto;
2

```

Data Output Messages Notifications

	nome_produto	documento_parceiro	sigla_estado	codigo_tabela_preco	cep_usuario	preco_produto	email_usuario
1	Kit Vidaria 500ml (Heisenberg L...	10.684.395/0001-70	PR	6000	85814-120	189.90	saulgoodman@gmail.co...
2	Kit Vidaria 500ml (Heisenberg L...	41.930.772/0001-08	PR	6000	85812-000	189.90	jessepinkman@gmail.c...
3	Kit Vidaria 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	saulgoodman@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	saulgoodman@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

Data Output Messages Notifications

Showing rows: 1 to 18 | Page No: 1 of 1 | < > | X

	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 Vidaria 500ml (Heisenberg L...	10.684.395/0001-70	PR	6000	85814-120	189,90	saulgoodman@gmail.co...
2	Kit Vidaria 500ml (Heisenberg L...	41.930.772/0001-08	PR	6000	85812-000	189,90	jessepinkman@gmail.c...
3	Kit Vidaria 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	saulgoodman@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	saulgoodman@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	saulgoodman@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	Galão Plástico Reforçado 20L	10.684.395/0001-70	PR	6000	85814-120	119,00	saulgoodman@gmail.co...
14	Galão Plástico Reforçado 20L	41.930.772/0001-08	PR	6000	85812-000	119,00	jessepinkman@gmail.c...
15	Galão 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	saulgoodman@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

Total rows: 18 Query complete 00:00:00.078

CRLF Ln 2, Col 1

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)
```

trabalho_bd/postgres@PostgreSQL 18

No limit

Query History

```
1 CREATE OR REPLACE FUNCTION calcular_imc(
2     p_peso NUMERIC,
3     p_altura NUMERIC
4 )
5 RETURNS NUMERIC AS $$ 
6 DECLARE
7     v_imc NUMERIC;
8 BEGIN
9     v_imc := p_peso / (p_altura * p_altura);
10    RETURN ROUND(v_imc, 2);
11 END;
12 $$ LANGUAGE plpgsql;
13
```

Scratch Pad

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 98 msec.

Total rows: Query complete 00:00:00.098

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

No limit

Query History

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

Scratch Pad

Query returned successfully in 98 msec. CRLF Ln 13, Col 1

Data Output Messages Notifications

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

calcular_imc	numeric
1	24.69

Successfully run. Total query runtime: 81 msec. 1 rows affected. CRLF Ln 2, Col 1

Total rows: 1 Query complete 00:00:00.081

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:

The screenshot shows the pgAdmin 4 interface with a query editor window. The title bar says "trabalho_bd/postgres@PostgreSQL 18". The query tab contains the following SQL code:

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

The status bar at the bottom shows "CREATE PROCEDURE" and "Query returned successfully in 93 msec.". A message box in the bottom right corner says "Query returned successfully in 93 msec. CRLF Ln 17, Col 1".

Print Testar:

The screenshot shows the pgAdmin 4 interface with a query editor window. The title bar says "trabalho_bd/postgres@PostgreSQL 18*". The query tab contains the following SQL code:

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

The status bar at the bottom shows "CALL" and "Query returned successfully in 70 msec.". A message box in the bottom right corner says "Query returned successfully in 70 msec. CRLF Ln 2, Col 1".

Print Confirmar:

The screenshot shows a PostgreSQL client interface with the following details:

- Connection:** trabalho_bd/postgres@PostgreSQL 18
- Query Tab:** Contains the SQL query:

```
1 SELECT idproduto, codigo, nome
2 FROM produto
3 WHERE codigo = 'AZ-HEIS-02';
4
```
- Data Output Tab:** Shows the results of the query in a table format:

	idproduto	codigo	nome
1	5001	AZ-HEIS-02	Balança Digital de Precisão 0.01g (Rev...
- Status Bar:** Includes messages like "Successfully run. Total query runtime: 83 msec. 1 rows affected.", "Total rows: 1", "Query complete 00:00:00.083", "CRLF", "Ln 4, Col 1", and "Page No: 1 of 1".

Print Stored Procedure para deletar produto pelo código:

The screenshot shows the pgAdmin 4 interface with a query editor window. The title bar indicates the connection is to 'trabalho_bd/postgres@PostgreSQL 18'. The main area contains the following PostgreSQL code:

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

Below the code, the status bar shows 'CREATE PROCEDURE' and 'Query returned successfully in 78 msec.' At the bottom, there are tabs for 'Data Output', 'Messages' (which is selected), and 'Notifications'. The footer displays 'Total rows: 0' and 'Query complete 00:00:00.078'.

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

The screenshot shows the pgAdmin 4 interface. The top window is a Query History tab containing the following SQL code:

```
1 INSERT INTO produto (codigo, nome, peso)
2 VALUES ('DEL-TEST-45', 'Produto para teste de deleção', 0.99);
3
4 INSERT INTO tabelapreproduto (idtabelapreco, idproduto, preco)
5 SELECT 6600, idproduto, 9.99
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
```

The bottom window is a Data Output tab showing the schema of the `produto` table:

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

At the bottom of the interface, there is a status bar with the message "Successfully run. Total query runtime: 76 msec. 0 rows affected." and "CRLF Ln 7, Col 30".

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:

The screenshot shows the pgAdmin 4 interface with a database connection named 'trabalho_bd/postgres@PostgreSQL 18'. In the 'Query' tab, a SQL script is run to create a table named 'pedido_audit'. The table structure includes fields for current values (e.g., idpedido_audit, idusuario_atual, idparceiro_atual, idtabelpreco_atual, idcondicaopagamento_atual, valortotal_atual, quantidadeprodutos_atual) and old values (e.g., idusuario_old, idparceiro_old, idtabelpreco_old, idcondicaopagamento_old, valortotal_old, quantidadeprodutos_old). It also includes columns for the last update timestamp ('ultima_atualizacao') and the type of operation ('ultima_operacao'). The query is successfully executed, returning 0 rows.

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

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 94 msec.

Total rows: 0 Query complete 00:00:00.094 ✓ Query returned successfully in 94 msec. × CRLF | Ln 23, Col 1

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

The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** Includes icons for file operations, search, and database navigation.
- Query History:** Shows the SQL code for creating the trigger function.
- Scratch Pad:** An empty tab for testing code.
- Data Output:** Shows the result of the query.
- Messages:** Displays a success message: "Query returned successfully in 65 msec."
- Notifications:** No notifications are present.
- Bottom Status Bar:** Shows "Total rows: 0" and "Query complete 00:00:00.065".
- Bottom Right:** A green status bar with a checkmark and the text "Query returned successfully in 65 msec. CRLF Ln 49, Col 1".

```
CREATE OR REPLACE FUNCTION fn_auditoria_pedido()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO pedido_audit (
            idusuario_atual, idparceiro_atual, idtabelapreco_atual, idcondicaopagamento_atual,
            valortotal_atual, quantidadeprodutos_atual,
            ultima_atualizacao, ultima_operacao
        )
        VALUES (
            NEW.idusuario, NEW.idparceiro, NEW.idtabelapreco, NEW.idcondicaopagamento,
            NEW.valortotal, NEW.quantidadeprodutos,
            NOW(), 'Insert'
        );
        RETURN NEW;
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO pedido_audit (
            idusuario_atual, idparceiro_atual, idtabelapreco_atual, idcondicaopagamento_atual,
            valortotal_atual, quantidadeprodutos_atual,
            idusuario_old, idparceiro_old, idtabelapreco_old, idcondicaopagamento_old,
            valortotal_old, quantidadeprodutos_old,
            ultima_atualizacao, ultima_operacao
        )
        VALUES (
            ...
        );
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Print Criar a trigger:

The screenshot shows a database query interface with two tabs: 'Query' and 'Scratch Pad'. The 'Query' tab contains the SQL code for creating a trigger:

```
1 CREATE TRIGGER trg_auditoria_pedido
2 AFTER INSERT OR UPDATE OR DELETE ON pedido
3 FOR EACH ROW
4 EXECUTE FUNCTION fn_auditoria_pedido();
```

The 'Messages' tab below shows the execution results:

```
CREATE TRIGGER
Query returned successfully in 66 msec.
```

At the bottom, status information includes 'Total rows: 0', 'Query complete 00:00:00.066', and a green message box stating 'Query returned successfully in 66 msec.' with a CRLF Ln 5, Col 1.

Print da auditoria funcionando corretamente para todas as operações possíveis:

The screenshot shows the pgAdmin 4 interface with a query editor and a results grid. The query editor contains a PL/pgSQL script that performs an insert, a delete, and an update on a 'pedido' table, followed by a select from an audit table. The results grid displays three rows of audit data corresponding to these operations. A status bar at the bottom indicates the query was successfully run.

```
25 INSERT INTO pedidoproduto (idpedido, idproduto, valorvenda)
26   VALUES (v_idpedido, 5002, 79.90);
27
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
42
43
44
45
46
47
48
49
```

idpedido_audit	[PK] integer	valortotal_atual	numeric (10,2)	valortotal_old	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		[null]		2026-01-23 02:12:28.681542		Insert	
2	3	169.90		149.90		2		1		2026-01-23 02:12:28.681542		Atualizado	
3	4	[null]		169.90		[null]		2		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:

The screenshot shows a PostgreSQL database interface with the following details:

- Connection:** trabalho_bd/postgres@PostgreSQL 18*
- Query Pad:** Contains the SQL code for creating a trigger function.

```
1 CREATE OR REPLACE FUNCTION fn_validar_pedido()
2 RETURNS TRIGGER AS $$
3 BEGIN
4     IF NEW.valortotal < 1 THEN
5         RAISE EXCEPTION 'Valor total não pode ser menor que 1';
6     END IF;
7
8     IF NEW.quantidadeprodutos < 0 THEN
9         RAISE EXCEPTION 'Quantidade de produtos não pode ser negativa';
10    END IF;
11
12    IF NEW.datapedido < CURRENT_DATE THEN
13        RAISE EXCEPTION 'Data do pedido não pode ser menor que a data atual';
14    END IF;
15
16    RETURN NEW;
17
18 END;
19 $$ LANGUAGE plpgsql;
```
- Data Output:** Shows the message "CREATE FUNCTION" and "Query returned successfully in 84 msec."
- Status Bar:** Total rows: 0, Query complete 00:00:00.084, CRLF, Ln 19, Col 1

Print Criar trigger de validação:

The screenshot shows the pgAdmin 4 interface for PostgreSQL 18. The top navigation bar includes Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, Processes, and the current connection information: trabalho_bd/postgres@PostgreSQL_18*.

The main window has two tabs: Query (selected) and Scratch Pad. The Query tab contains the following SQL code:

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

The Messages tab shows the execution results:

```
CREATE TRIGGER
Query returned successfully in 64 msec.
```

The status bar at the bottom indicates Total rows: 0, Query complete 00:00:00.064, and a green message box stating "Query returned successfully in 64 msec." with a CRLF Ln 5, Col 1.

Print Testes de erro

Valor inválido:

The screenshot shows a PostgreSQL query editor interface. At the top, there are tabs for 'Query' (which is selected) and 'Query History'. Below the tabs, the SQL query is displayed:

```
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
```

At the bottom of the editor, there are three tabs: 'Data Output', 'Messages' (which is selected), and 'Notifications'. The 'Messages' tab displays the following error output:

```
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  INSERT INTO pedido
2  (datapedido, valortotal, quantidadeprodutos, idusuario, idparceiro, idtabelapreco, idcondicaopagamento)
3  VALUES
4  (CURRENT_DATE, 100, -1, 1000, 2000, 6000, 7000);
5

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

ERRO: Data do pedido não pode ser menor que a data atual
SQL state: P0001