

---

# Banco de Dados – IMD0401

## Aula 21 – Visão e Gatilho

---

João Carlos Xavier Júnior

[jcxavier@imd.ufrn.br](mailto:jcxavier@imd.ufrn.br)

# DDL – view

## ❑ Conceito:

- ❖ Uma visão (ou view) é uma **relação virtual**, que não faz parte do modelo lógico, mas que é visível a um grupo de usuários.
- ❖ A visão é definida por uma **DDL**, e é computada cada vez que consultas são realizadas aos dados daquela visão.
- ❖ O SGBD armazena as definições das visões no **Dicionário de Dados** (DD ou Metadados).
- ❖ Uma visão possui um **nome**, uma **seleção** e uma **projeção de atributos**.
- ❖ As consultas são acessadas através de consultas SQL.

# DDL – view

## ❑ Conceito:

- ❖ As visões são acessadas através de consultas SQL.



[http://pt.123rf.com/photo\\_28263454\\_c%C3%B3digo-de-sintaxe-sql-no-fundo-branco.html](http://pt.123rf.com/photo_28263454_c%C3%B3digo-de-sintaxe-sql-no-fundo-branco.html)

# DDL – view

## ❑ Conceito:

- ❖ A visão é definida a partir de **tabelas do banco**, contendo sempre os dados atualizados.
- ❖ Visões **não são cópias** separadas dos dados.
  - Elas fazem referência aos dados de tabelas existentes no banco de dados.
  - São massas de dados virtuais.

---

# DDL – view

## □ Conceito:

### ❖ Vantagens:

- Maneira simples de executar e exibir dados;
- Economizar tempo com retrabalho;
- Velocidade de acesso às informações.

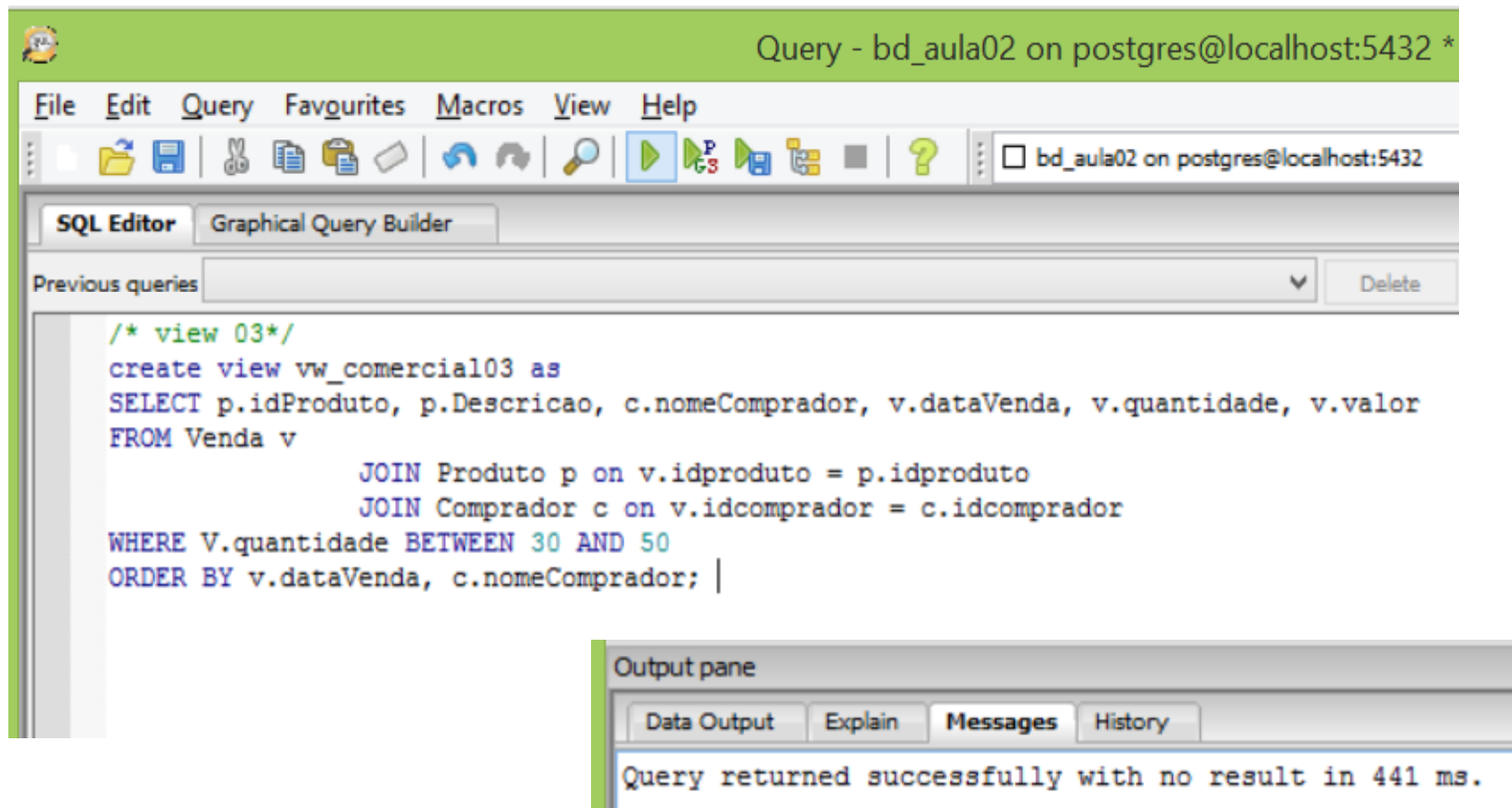
# DDL – view

## ❑ Codificando uma view:

```
/* view 03*/  
create view vw_comercial03 as  
SELECT p.idProduto, p.Descricao, c.nomeComprador,  
v.dataVenda, v.quantidade, v.valor  
FROM Venda v  
  
JOIN Produto p on v.idproduto = p.idproduto  
JOIN Comprador c on v.idcomprador = c.idcomprador  
  
WHERE V.quantidade BETWEEN 30 AND 50  
ORDER BY v.dataVenda, c.nomeComprador;
```

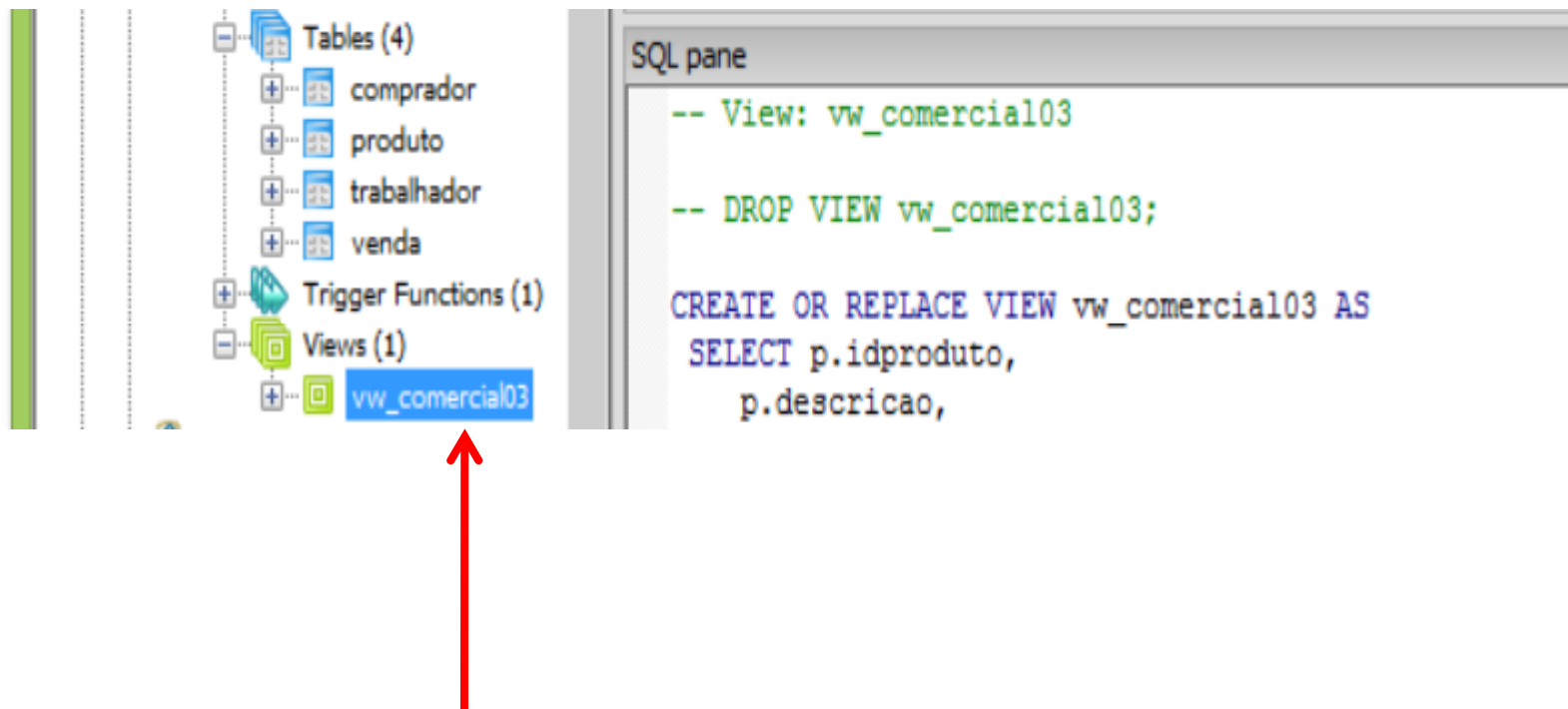
# DDL – view

## ❑ Criando uma view:



# DDL – view

## ❑ Criando uma view:



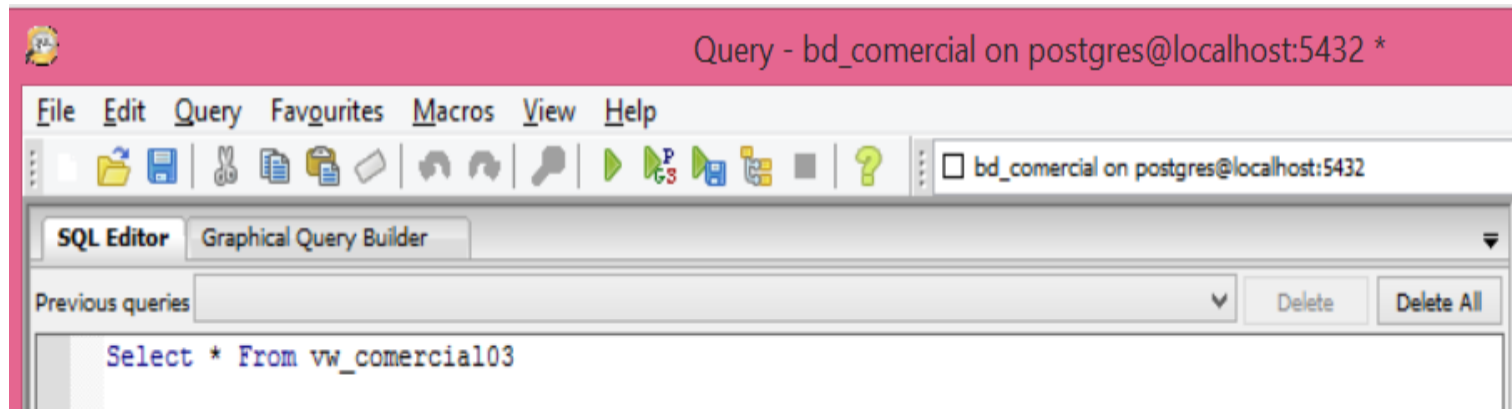
The screenshot displays a database management interface. On the left, a tree view shows the database structure with categories: Tables (4), Trigger Functions (1), and Views (1). The 'Views (1)' category is expanded, and the view 'vw\_comercial03' is highlighted. A red arrow points to this view. On the right, the 'SQL pane' contains the following SQL code:

```
-- View: vw_comercial03  
  
-- DROP VIEW vw_comercial03;  
  
CREATE OR REPLACE VIEW vw_comercial03 AS  
SELECT p.idproduto,  
       p.descricao,
```



# DDL – view

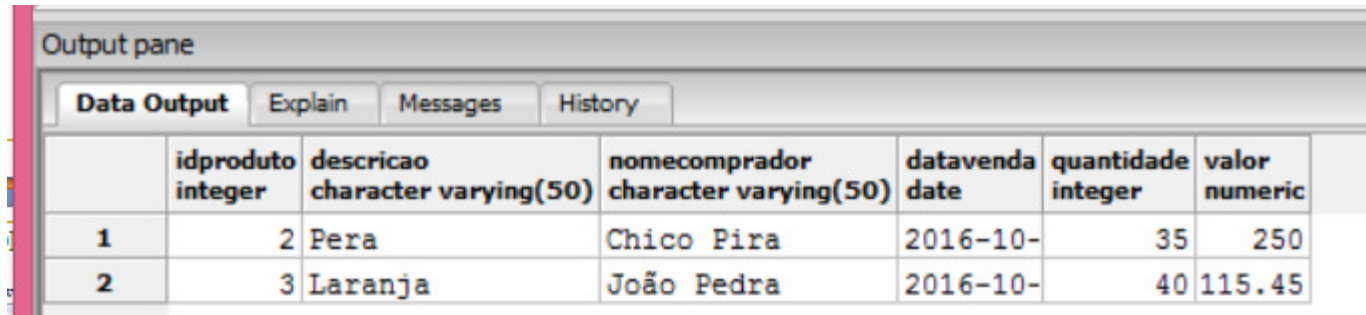
## ❑ Executando uma view:



**Select** \* **From** vw\_comercial03

# DDL – view

## ❑ Resultado:

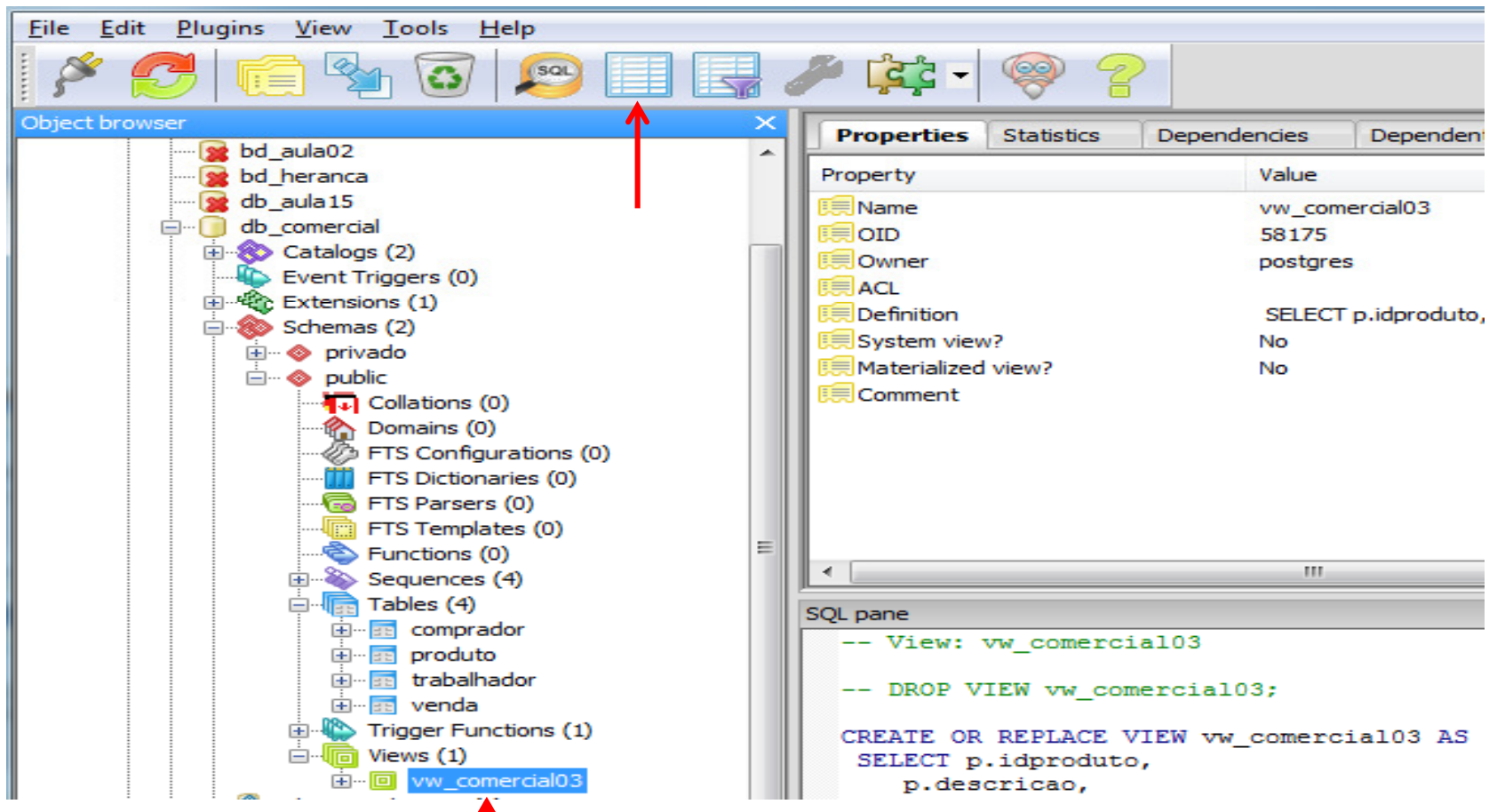


The screenshot shows a database interface with an 'Output pane' at the top. Below the pane title are four tabs: 'Data Output', 'Explain', 'Messages', and 'History'. The 'Data Output' tab is selected, displaying a table with 7 columns and 2 data rows. The columns are: an unlabeled index column, 'idproduto' (integer), 'descricao' (character varying(50)), 'nomecomprador' (character varying(50)), 'datavenda' (date), 'quantidade' (integer), and 'valor' (numeric). The first row shows index 1, id 2, description 'Pera', buyer 'Chico Pira', date '2016-10-', quantity 35, and value 250. The second row shows index 2, id 3, description 'Laranja', buyer 'João Pedra', date '2016-10-', quantity 40, and value 115.45.

	idproduto integer	descricao character varying(50)	nomecomprador character varying(50)	datavenda date	quantidade integer	valor numeric
1	2	Pera	Chico Pira	2016-10-	35	250
2	3	Laranja	João Pedra	2016-10-	40	115.45

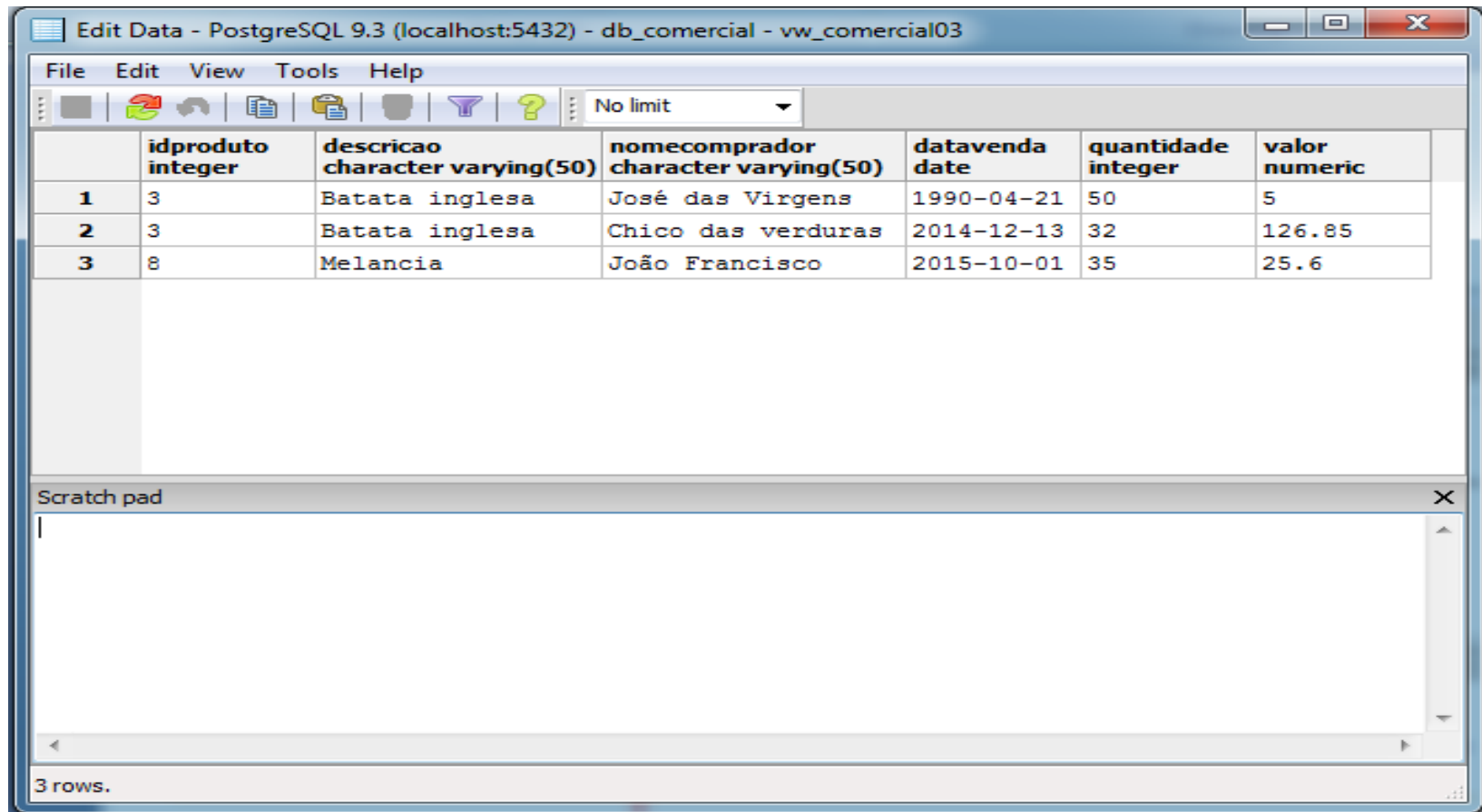
# DDL – view

## ❑ Executando uma view:



# DDL – view

## ❑ Executando uma view:



The screenshot shows a PostgreSQL 9.3 'Edit Data' window for a database named 'db\_comercial' and a view named 'vw\_comercial03'. The window displays a table with 7 columns: 'idproduto' (integer), 'descricao' (character varying(50)), 'nomecomprador' (character varying(50)), 'datavenda' (date), 'quantidade' (integer), and 'valor' (numeric). There are 3 rows of data. Below the table is a 'Scratch pad' area. The status bar at the bottom indicates '3 rows.'.

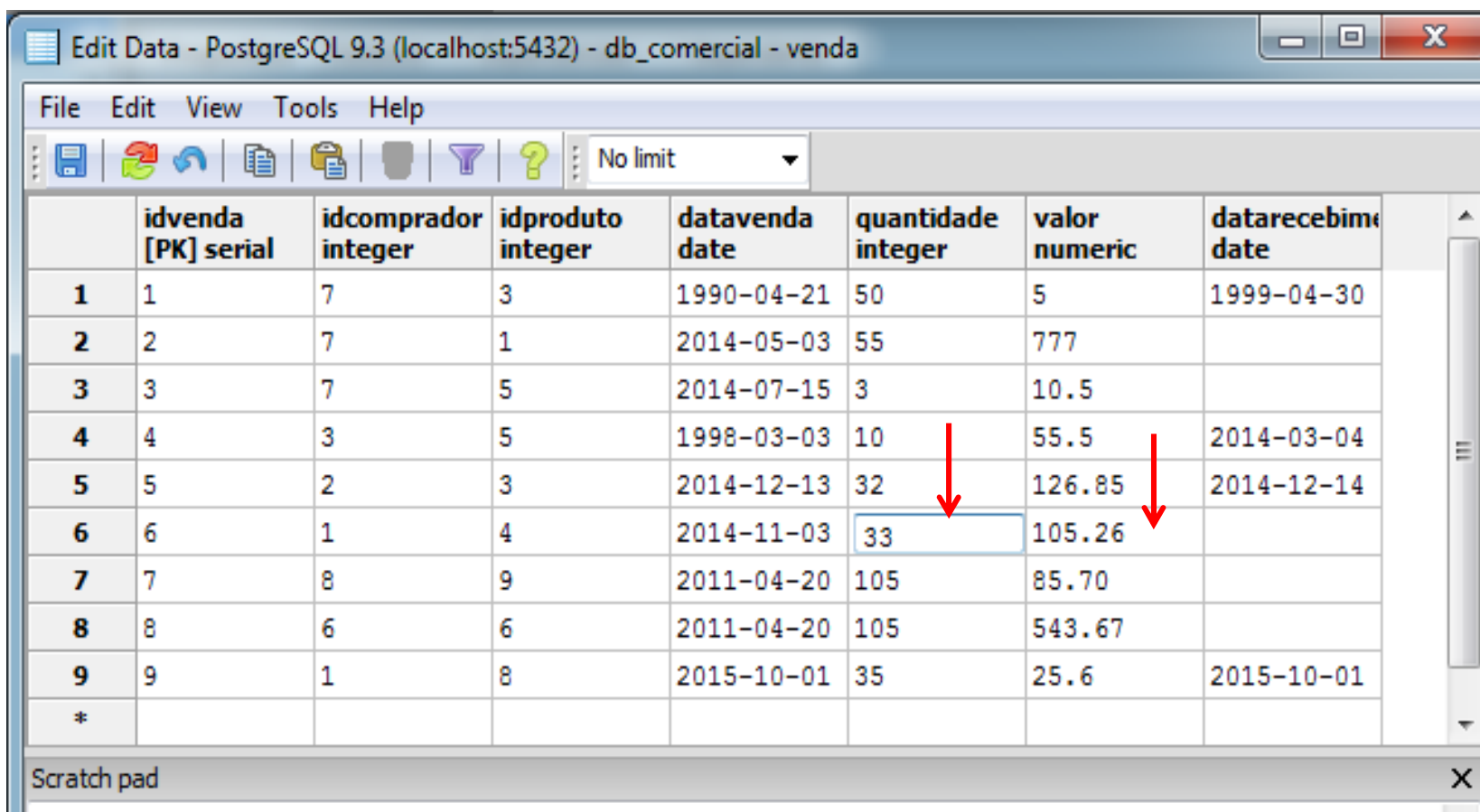
	<b>idproduto</b> integer	<b>descricao</b> character varying(50)	<b>nomecomprador</b> character varying(50)	<b>datavenda</b> date	<b>quantidade</b> integer	<b>valor</b> numeric
1	3	Batata inglesa	José das Virgens	1990-04-21	50	5
2	3	Batata inglesa	Chico das verduras	2014-12-13	32	126.85
3	8	Melancia	João Francisco	2015-10-01	35	25.6

Scratch pad

3 rows.

# SQL - DQL

## ❑ Alterando os dados:

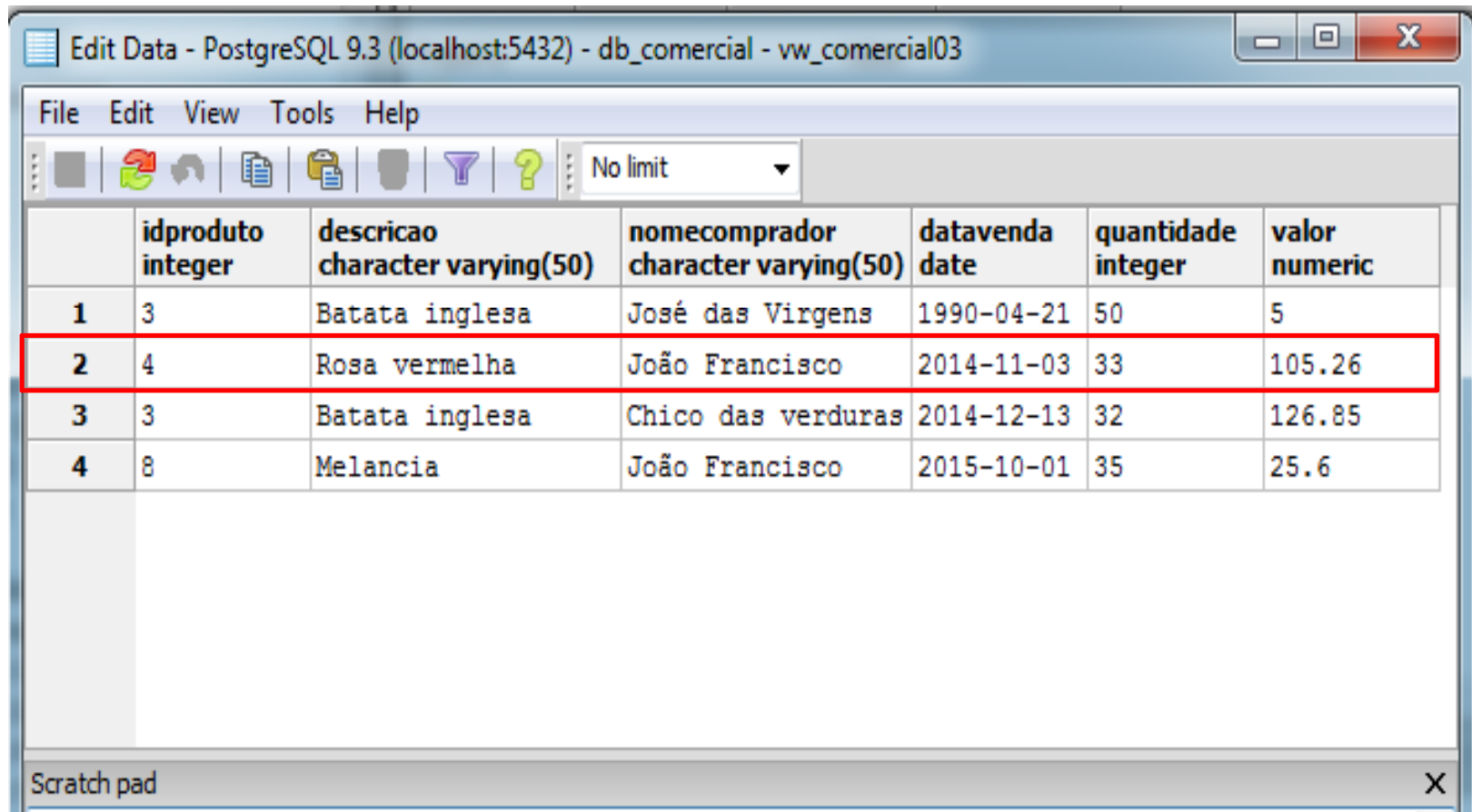


Scratch pad

	idvenda [PK] serial	idcomprador integer	idproduto integer	datavenda date	quantidade integer	valor numeric	datarecebime date
1	1	7	3	1990-04-21	50	5	1999-04-30
2	2	7	1	2014-05-03	55	777	
3	3	7	5	2014-07-15	3	10.5	
4	4	3	5	1998-03-03	10	55.5	2014-03-04
5	5	2	3	2014-12-13	32	126.85	2014-12-14
6	6	1	4	2014-11-03	33	105.26	
7	7	8	9	2011-04-20	105	85.70	
8	8	6	6	2011-04-20	105	543.67	
9	9	1	8	2015-10-01	35	25.6	2015-10-01
*							

# SQL - DQL

## □ Atualizando a view:



PostgreSQL 9.3 (localhost:5432) - db\_comercial - vw\_comercial03

File Edit View Tools Help

No limit

	idproduto integer	descricao character varying(50)	nomecomprador character varying(50)	datavenda date	quantidade integer	valor numeric
1	3	Batata inglesa	José das Virgens	1990-04-21	50	5
2	4	Rosa vermelha	João Francisco	2014-11-03	33	105.26
3	3	Batata inglesa	Chico das verduras	2014-12-13	32	126.85
4	8	Melancia	João Francisco	2015-10-01	35	25.6

Scratch pad

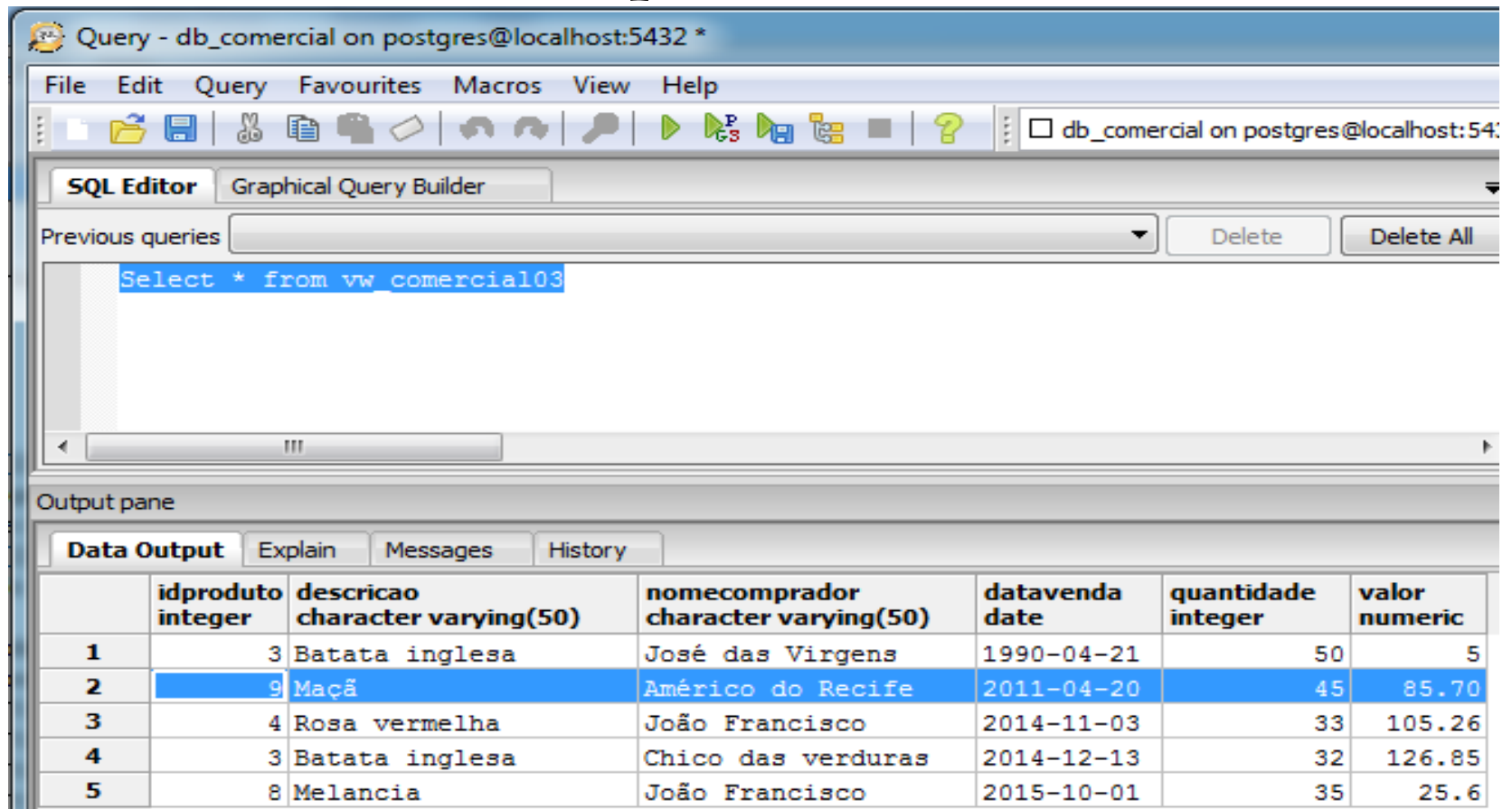
# SQL - DQL

## ❑ Inserindo mais dados:

```
/* inserção de produto, comprador e venda */  
INSERT INTO produto( descricao, areaPlantada, idTrabalhador)  
VALUES ('Maçã', 15.2, 4);  
INSERT INTO comprador( nomeComprador, cidadeComprador, telefoneComprador)  
VALUES ('Américo do Recife', 'Recife', '9874-9191');  
INSERT INTO venda (idComprador, idProduto, dataVenda, quantidade, valor)  
VALUES (8, 9, '20-04-2011', 45, 85.70);
```

# SQL - DQL

- ❑ Atualizando a view depois da inserção:



The screenshot shows a PostgreSQL SQL Editor window titled "Query - db\_comercial on postgres@localhost:5432 \*". The window has a menu bar (File, Edit, Query, Favourites, Macros, View, Help) and a toolbar with various icons. Below the toolbar, there are tabs for "SQL Editor" and "Graphical Query Builder". The SQL Editor tab is active, showing a query: `Select * from vw_comercial03`. Below the query editor, there is an "Output pane" with tabs for "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is active, displaying a table with 7 columns: `idproduto` (integer), `descricao` (character varying(50)), `nomecomprador` (character varying(50)), `datavenda` (date), `quantidade` (integer), and `valor` (numeric). The table contains 5 rows of data, with the second row highlighted in blue.

	<code>idproduto</code> integer	<code>descricao</code> character varying(50)	<code>nomecomprador</code> character varying(50)	<code>datavenda</code> date	<code>quantidade</code> integer	<code>valor</code> numeric
1	3	Batata inglesa	José das Virgens	1990-04-21	50	5
2	9	Maçã	Américo do Recife	2011-04-20	45	85.70
3	4	Rosa vermelha	João Francisco	2014-11-03	33	105.26
4	3	Batata inglesa	Chico das verduras	2014-12-13	32	126.85
5	8	Melancia	João Francisco	2015-10-01	35	25.6



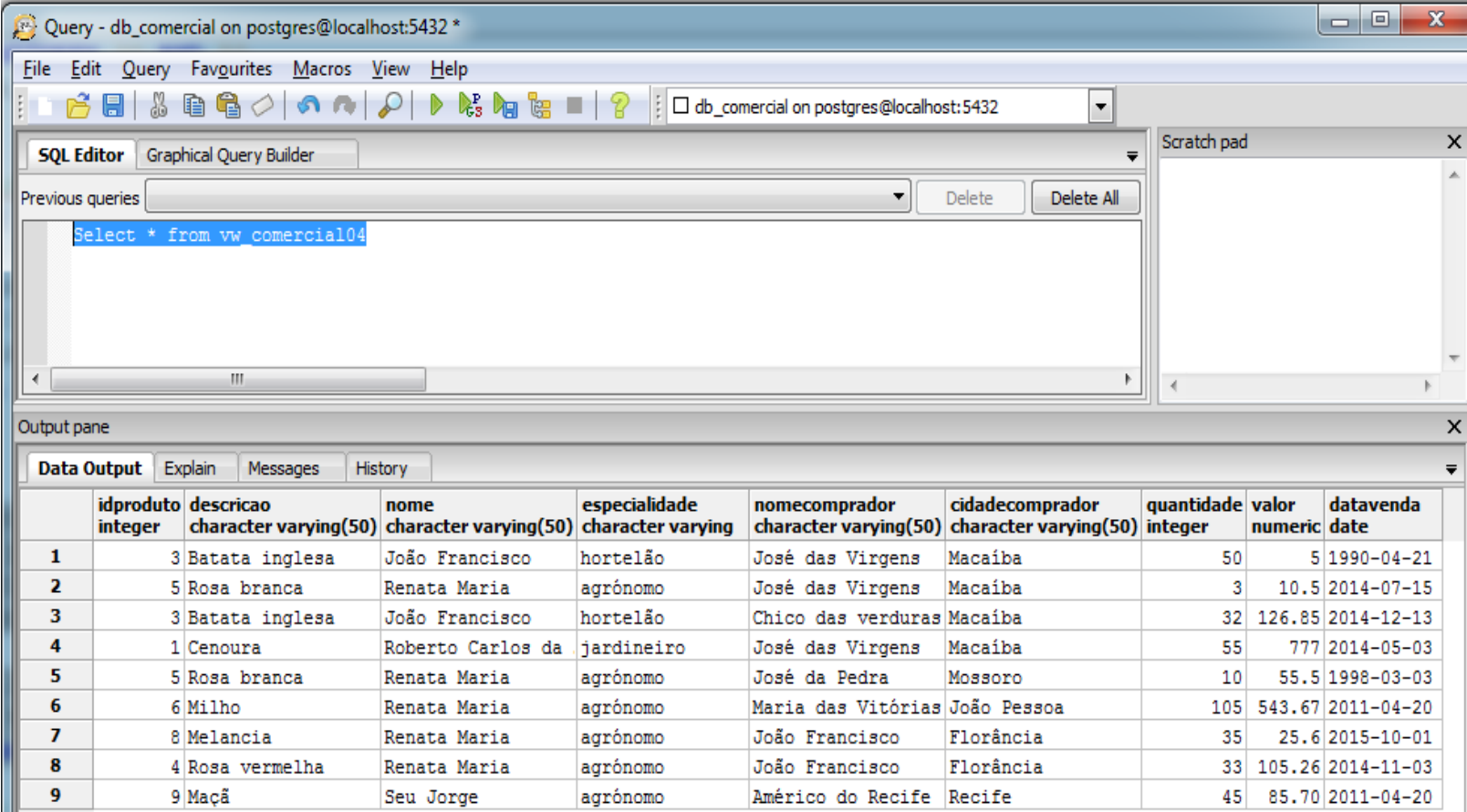
# DDL – view

## ❑ Outro exemplo de View:

```
/* view 04*/  
create view vw_comercial04 as  
SELECT p.idProduto, p.descricao, t.nome, t.especialidade,  
c.nomeComprador, c.cidadeComprador,  
v.quantidade, v.valor, v.dataVenda  
FROM Venda v  
JOIN Produto p on v.idproduto = p.idproduto  
JOIN Comprador c on v.idcomprador = c.idcomprador  
JOIN Trabalhador t on p.idTrabalhador = t.idTrabalhador
```

# DDL – view

## ❑ Executando vw\_comercial04:

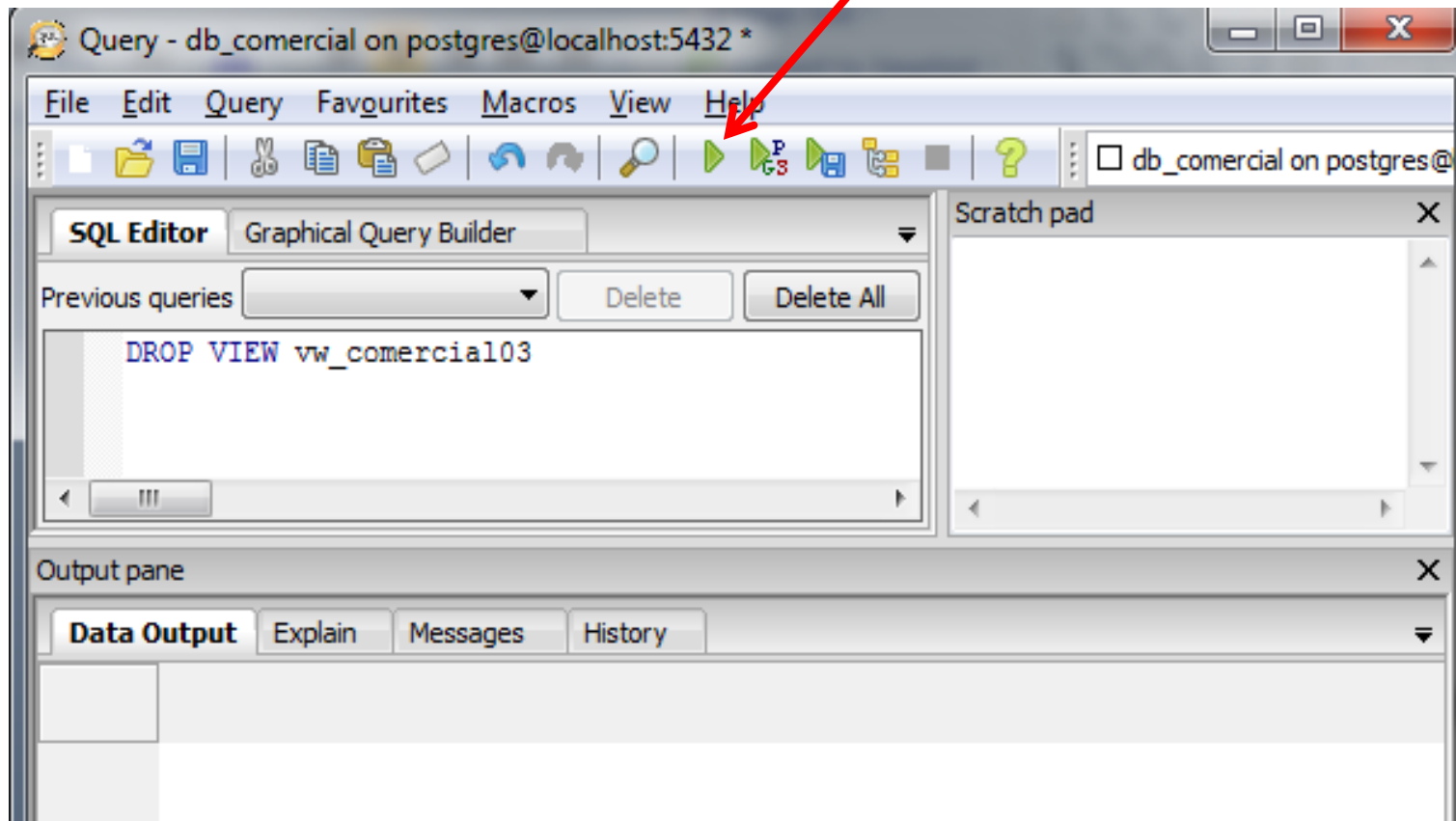


The screenshot shows a PostgreSQL query editor window titled "Query - db\_comercial on postgres@localhost:5432 \*". The window has a menu bar (File, Edit, Query, Favourites, Macros, View, Help) and a toolbar. The "SQL Editor" tab is active, displaying the query: `Select * from vw_comercial04`. The "Output pane" at the bottom shows the results of the query in a table format. The table has 10 columns: `idproduto` (integer), `descricao` (character varying(50)), `nome` (character varying(50)), `especialidade` (character varying), `nomecomprador` (character varying(50)), `cidadecomprador` (character varying(50)), `quantidade` (integer), `valor` (numeric), and `datavenda` (date). The results are displayed in a table with 9 rows of data.

	<code>idproduto</code> integer	<code>descricao</code> character varying(50)	<code>nome</code> character varying(50)	<code>especialidade</code> character varying	<code>nomecomprador</code> character varying(50)	<code>cidadecomprador</code> character varying(50)	<code>quantidade</code> integer	<code>valor</code> numeric	<code>datavenda</code> date
1	3	Batata inglesa	João Francisco	hortelão	José das Virgens	Macaíba	50	5	1990-04-21
2	5	Rosa branca	Renata Maria	agrônomo	José das Virgens	Macaíba	3	10.5	2014-07-15
3	3	Batata inglesa	João Francisco	hortelão	Chico das verduras	Macaíba	32	126.85	2014-12-13
4	1	Cenoura	Roberto Carlos da	jardineiro	José das Virgens	Macaíba	55	777	2014-05-03
5	5	Rosa branca	Renata Maria	agrônomo	José da Pedra	Mossoro	10	55.5	1998-03-03
6	6	Milho	Renata Maria	agrônomo	Maria das Vitórias	João Pessoa	105	543.67	2011-04-20
7	8	Melancia	Renata Maria	agrônomo	João Francisco	Florância	35	25.6	2015-10-01
8	4	Rosa vermelha	Renata Maria	agrônomo	João Francisco	Florância	33	105.26	2014-11-03
9	9	Maçã	Seu Jorge	agrônomo	Américo do Recife	Recife	45	85.70	2011-04-20

# DDL – view

## ❑ Apagando uma view:



---

# DDL – trigger

## □ Conceito:

- ❖ Um **gatilho** (ou **trigger**) é uma tarefa executada implicitamente sempre que um evento particular ocorre no banco de dados.
- ❖ Um **evento** pode ser inclusão, alteração ou exclusão de um **registro**.
- ❖ Usado para implementar **regras de negócios** no banco de dados.

# DDL – trigger

## ❑ Sintaxe PostgreSQL:

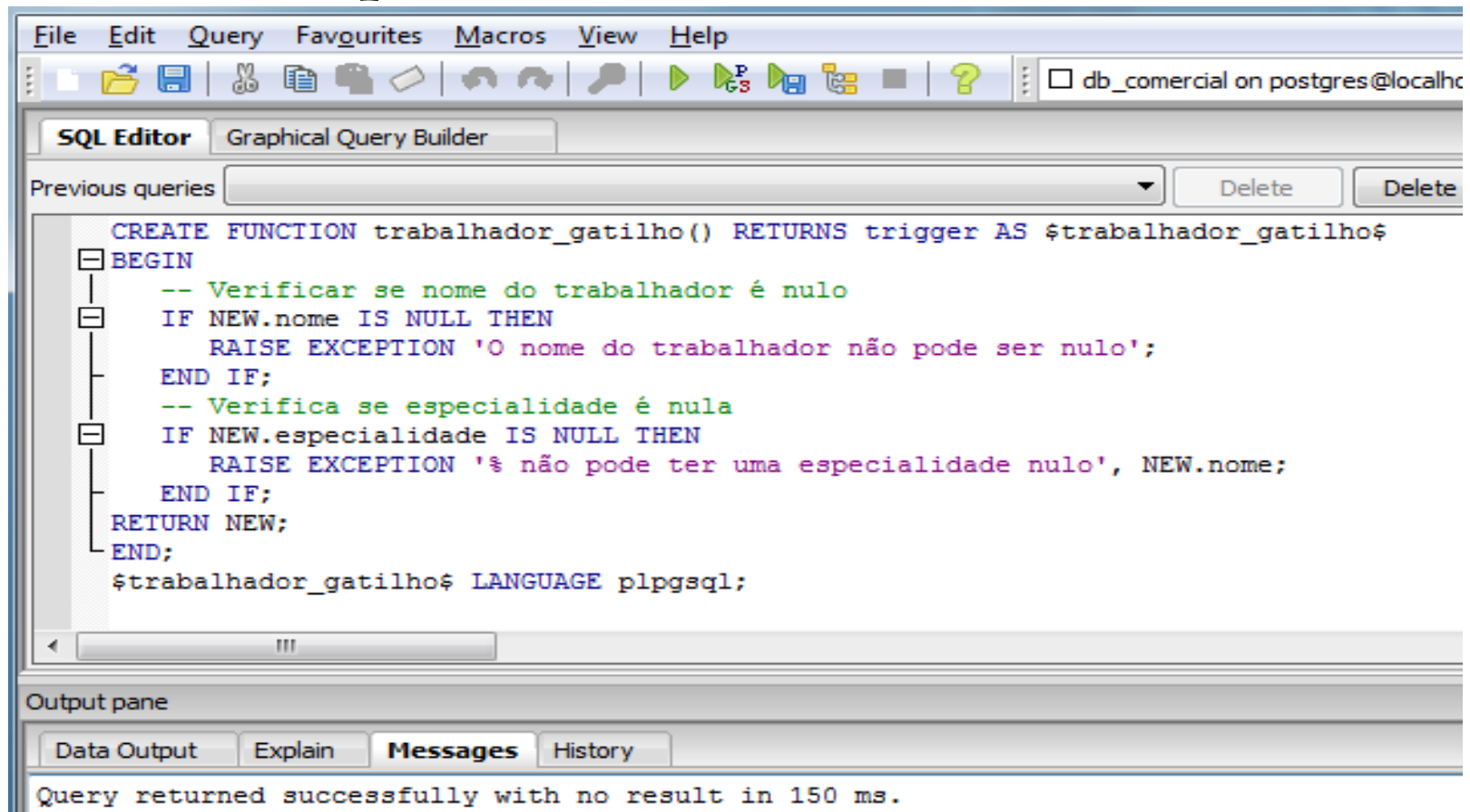
```
CREATE TRIGGER nome { BEFORE | AFTER } {  
evento [ OR ... ] }  
ON tabela [ FOR [ EACH ] { ROW | STATEMENT  
} ]  
EXECUTE PROCEDURE nome_da_função (  
argumentos )
```

## ❑ Observação:

- ❖ Para que o gatilho funcione, precisa que uma função seja definida em *pgpsql*.

# DDL – trigger

## ❑ Criando e especificando uma função:



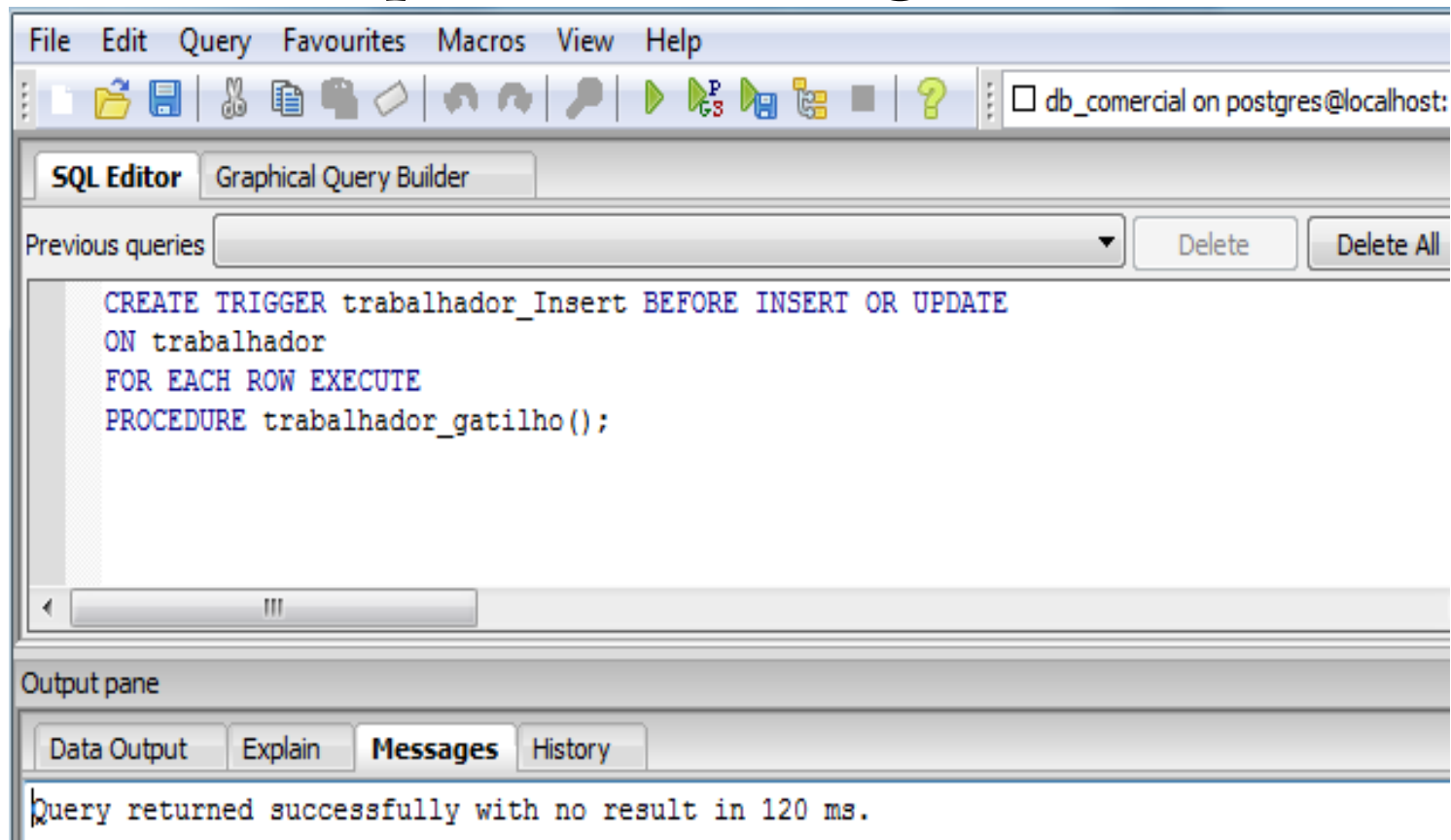
The screenshot shows a SQL Editor window with a menu bar (File, Edit, Query, Favouirites, Macros, View, Help) and a toolbar. The title bar indicates the connection is 'db\_comercial on postgres@localho'. The editor has two tabs: 'SQL Editor' (active) and 'Graphical Query Builder'. Below the tabs is a 'Previous queries' dropdown and two 'Delete' buttons. The main text area contains the following SQL code:

```
CREATE FUNCTION trabalhador_gatilho() RETURNS trigger AS $trabalhador_gatilho$
BEGIN
    -- Verificar se nome do trabalhador é nulo
    IF NEW.nome IS NULL THEN
        RAISE EXCEPTION 'O nome do trabalhador não pode ser nulo';
    END IF;
    -- Verifica se especialidade é nula
    IF NEW.especialidade IS NULL THEN
        RAISE EXCEPTION '% não pode ter uma especialidade nulo', NEW.nome;
    END IF;
    RETURN NEW;
END;
$trabalhador_gatilho$ LANGUAGE plpgsql;
```

Below the editor is an 'Output pane' with tabs for 'Data Output', 'Explain', 'Messages' (selected), and 'History'. The 'Messages' tab shows the message: 'Query returned successfully with no result in 150 ms.'

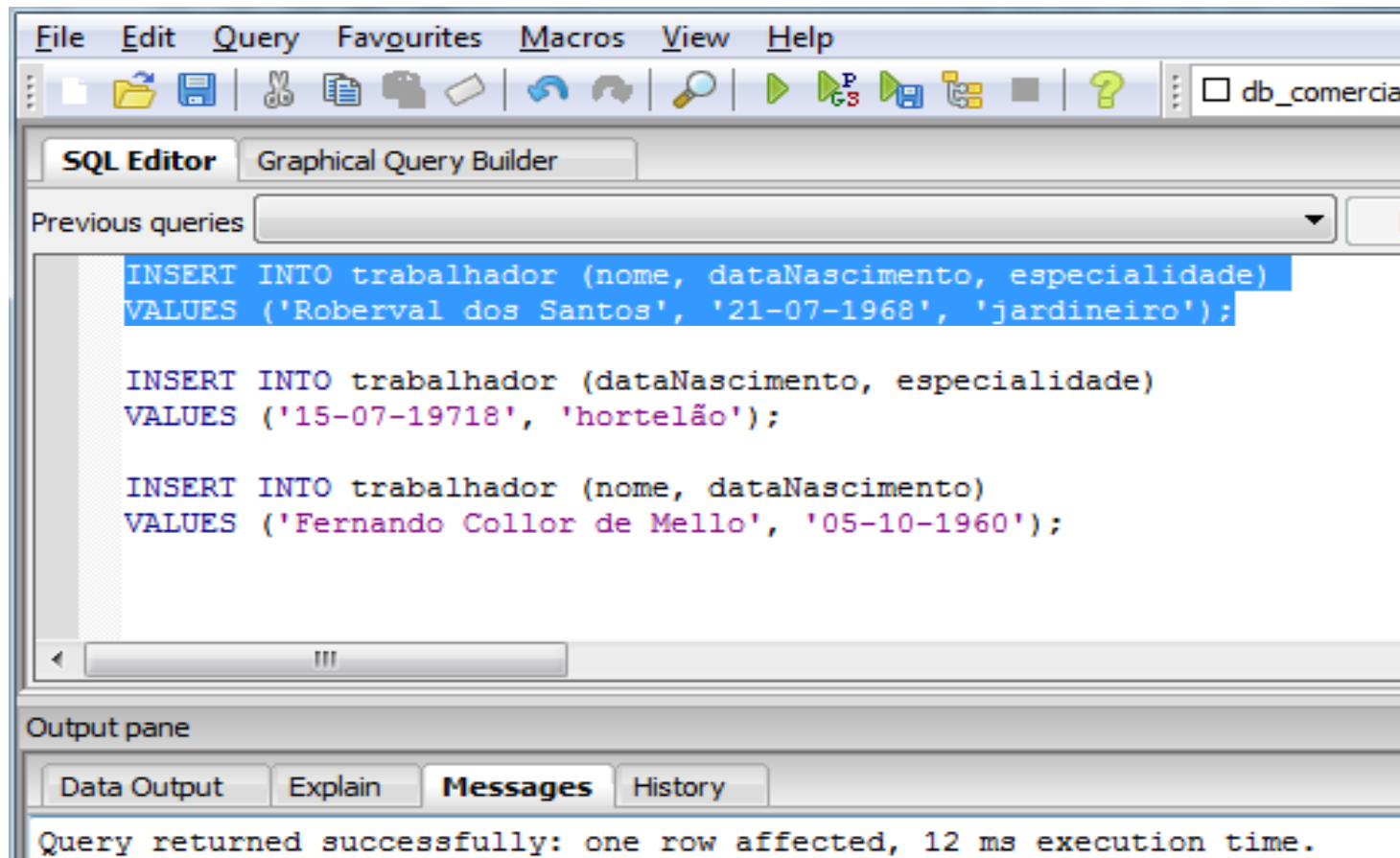
# DDL – trigger

## ❑ Criando e especificando um gatilho:



# DDL – trigger

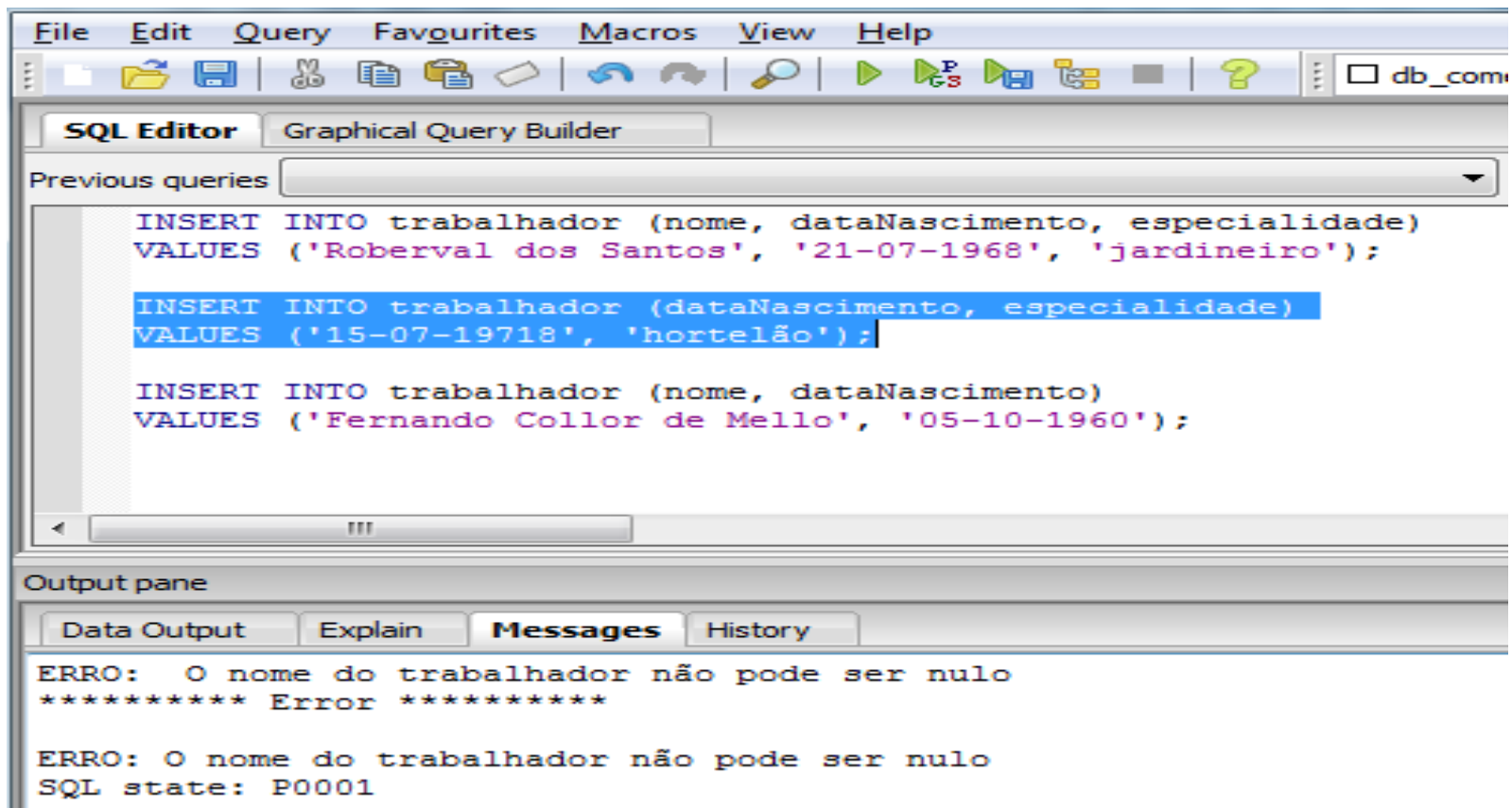
## ❑ Testando o funcionamento de uma função:





# DDL – trigger

- ❑ Testando o funcionamento de uma função:



The screenshot shows a SQL Editor window with a menu bar (File, Edit, Query, Favourites, Macros, View, Help) and a toolbar. The 'SQL Editor' tab is active. Below the menu bar is a 'Previous queries' dropdown. The main text area contains three SQL statements:

```
INSERT INTO trabalhador (nome, dataNascimento, especialidade)
VALUES ('Roberval dos Santos', '21-07-1968', 'jardineiro');

INSERT INTO trabalhador (dataNascimento, especialidade)
VALUES ('15-07-19718', 'hortelão');

INSERT INTO trabalhador (nome, dataNascimento)
VALUES ('Fernando Collor de Mello', '05-10-1960');
```

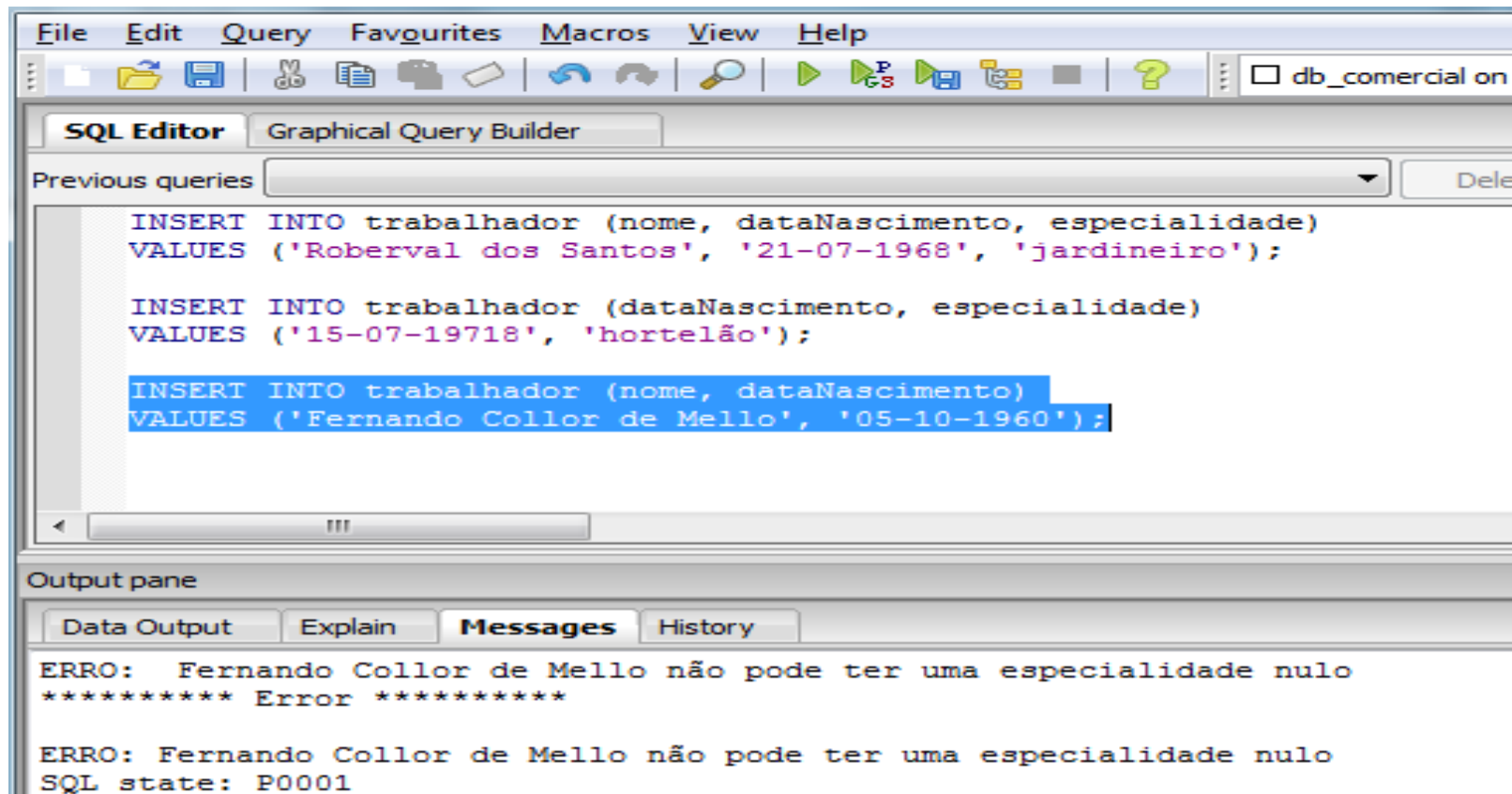
The second statement is highlighted in blue. Below the text area is a scrollbar. At the bottom is an 'Output pane' with tabs for 'Data Output', 'Explain', 'Messages', and 'History'. The 'Messages' tab is selected, showing two error messages:

```
ERRO: O nome do trabalhador não pode ser nulo
***** Error *****

ERRO: O nome do trabalhador não pode ser nulo
SQL state: P0001
```

# DDL – trigger

- ❑ Testando o funcionamento de uma função:



The screenshot shows a SQL Editor window with a menu bar (File, Edit, Query, Favourites, Macros, View, Help) and a toolbar. The 'SQL Editor' tab is active. The 'Previous queries' dropdown is empty. The SQL text area contains three INSERT statements for a table named 'trabalhador'. The third statement is highlighted in blue. The 'Output pane' at the bottom has tabs for 'Data Output', 'Explain', 'Messages', and 'History'. The 'Messages' tab is selected, displaying two error messages.

```
INSERT INTO trabalhador (nome, dataNascimento, especialidade)
VALUES ('Roberval dos Santos', '21-07-1968', 'jardineiro');

INSERT INTO trabalhador (dataNascimento, especialidade)
VALUES ('15-07-19718', 'hortelão');

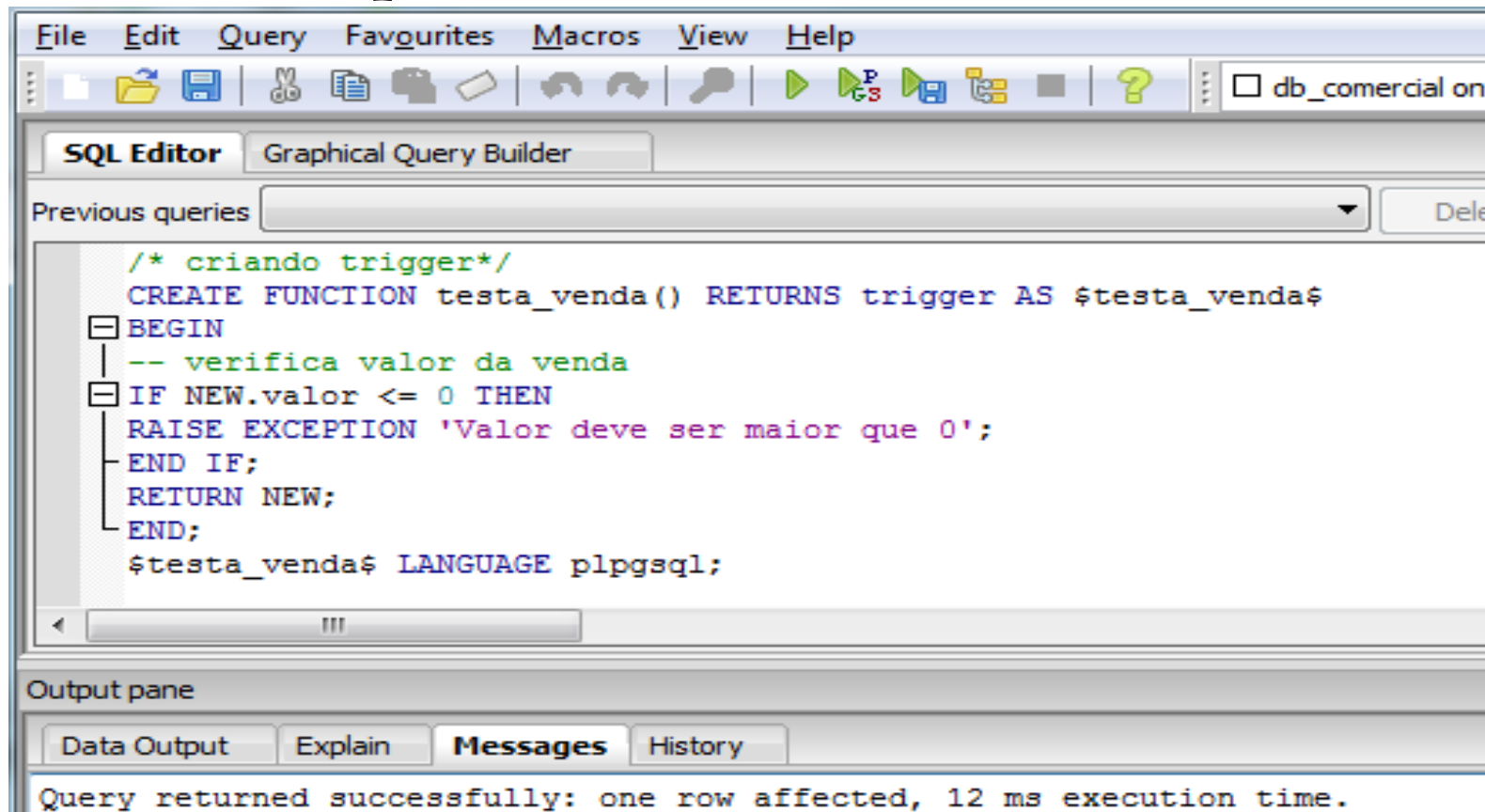
INSERT INTO trabalhador (nome, dataNascimento)
VALUES ('Fernando Collor de Mello', '05-10-1960');
```

ERRO: Fernando Collor de Mello não pode ter uma especialidade nulo  
\*\*\*\*\* Error \*\*\*\*\*

ERRO: Fernando Collor de Mello não pode ter uma especialidade nulo  
SQL state: P0001

# DDL – Trigger

- ❑ Criando e especificando uma outra função:



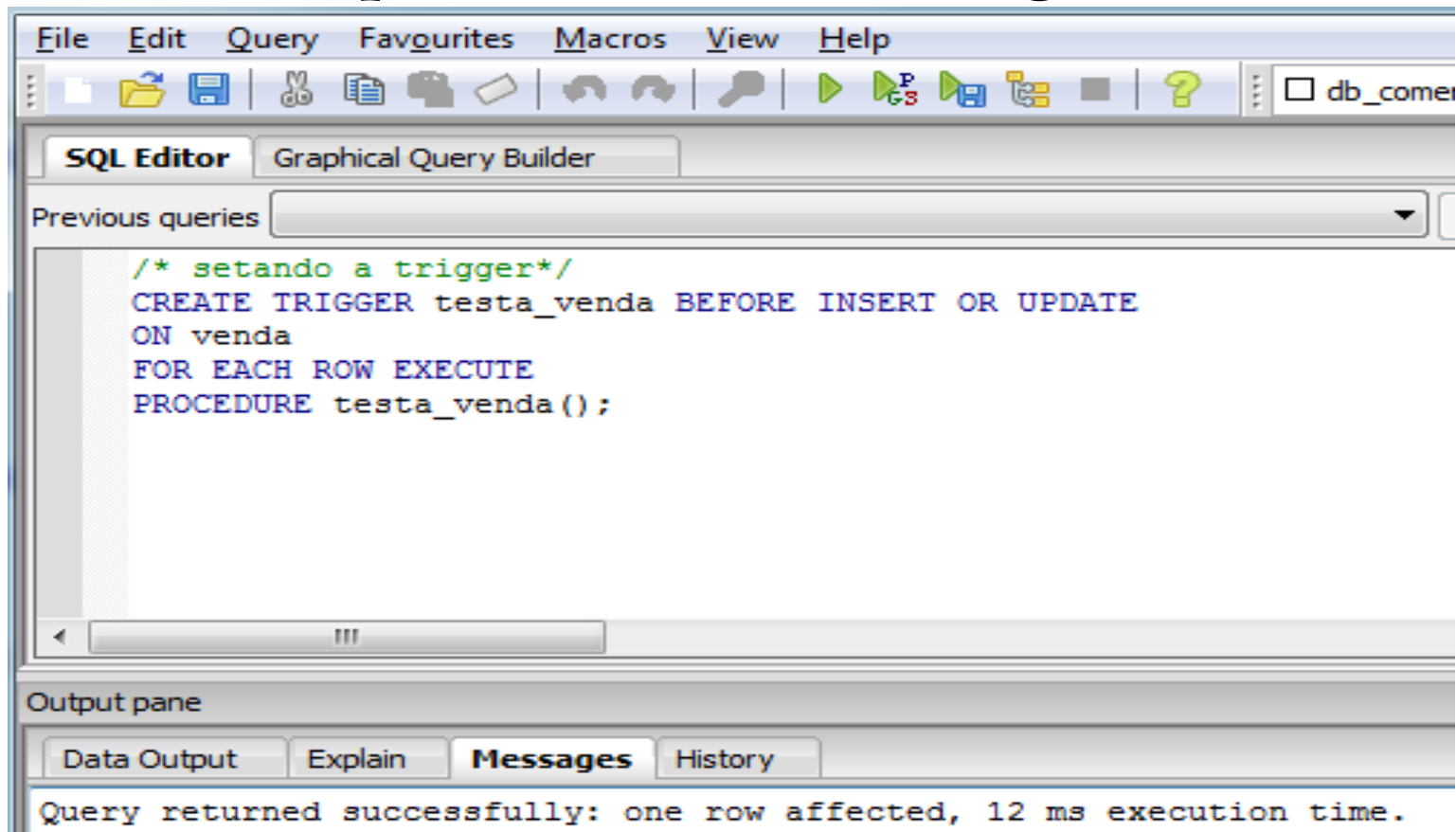
The screenshot shows an SQL Editor window with a menu bar (File, Edit, Query, Favouirites, Macros, View, Help) and a toolbar. The 'SQL Editor' tab is active. The main text area contains the following SQL code:

```
/* criando trigger*/  
CREATE FUNCTION testa_venda() RETURNS trigger AS $testa_venda$  
BEGIN  
  -- verifica valor da venda  
  IF NEW.valor <= 0 THEN  
    RAISE EXCEPTION 'Valor deve ser maior que 0';  
  END IF;  
  RETURN NEW;  
END;  
$testa_venda$ LANGUAGE plpgsql;
```

Below the code editor is an 'Output pane' with tabs for 'Data Output', 'Explain', 'Messages', and 'History'. The 'Messages' tab is selected, displaying the message: 'Query returned successfully: one row affected, 12 ms execution time.'

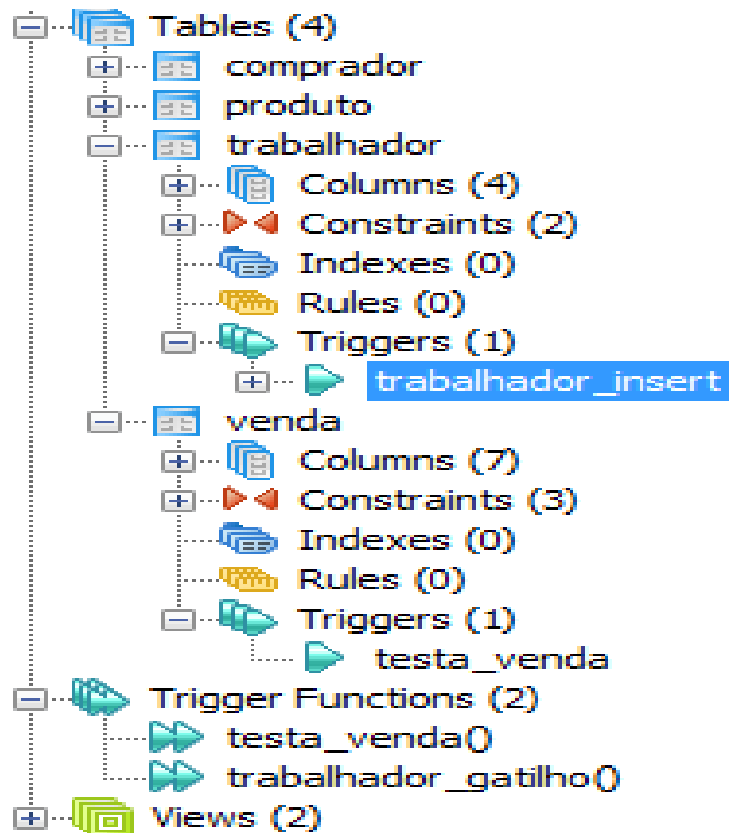
# DDL – Trigger

- ❑ Criando e especificando um outro gatilho:



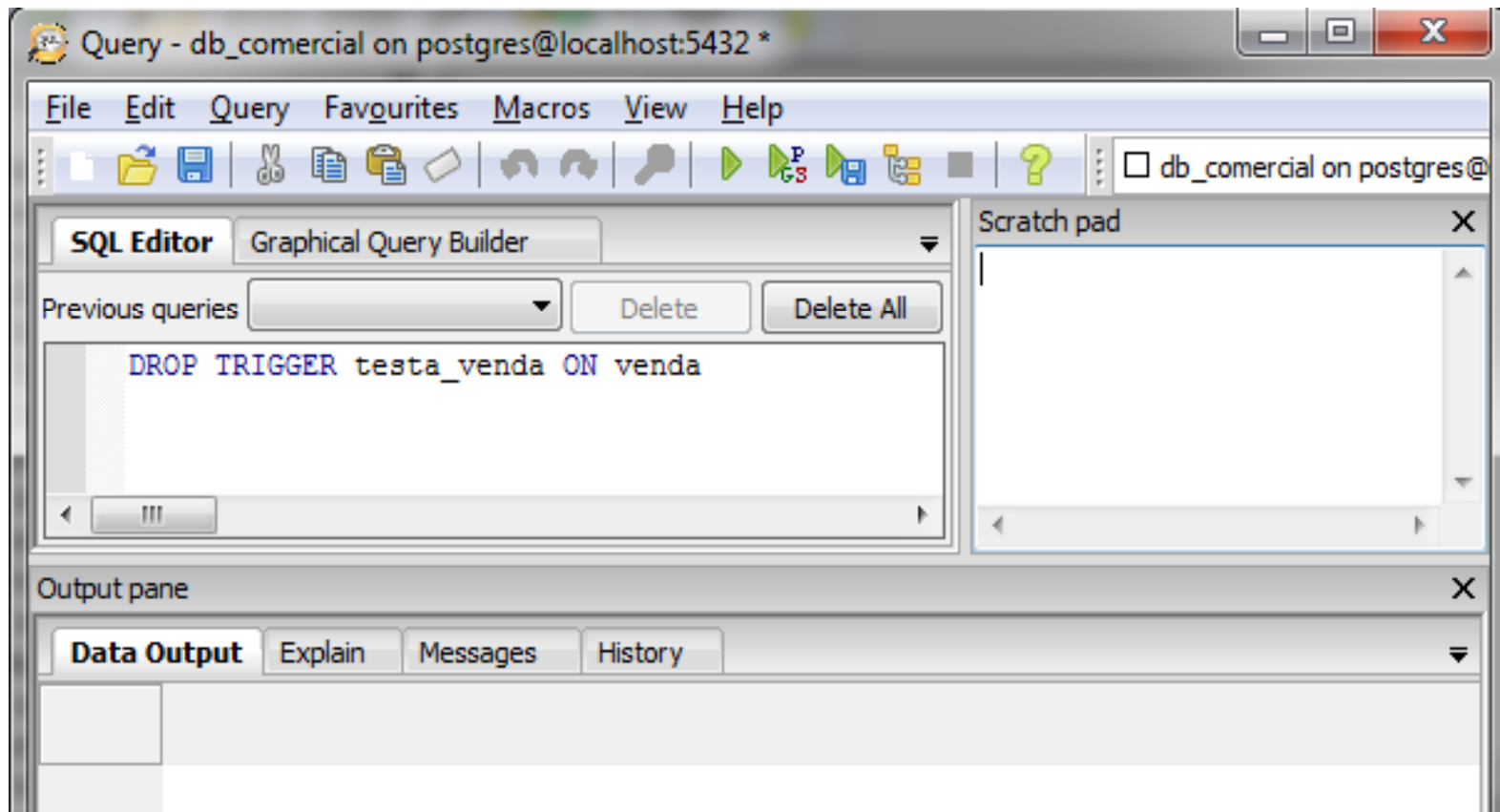
# DDL – Trigger

## □ Funções e gatilhos criados:



# DDL – Trigger

## ❑ Dropando (DROP) um Gatilho:



# Dúvidas...



“O conhecimento é orgulhoso por ter aprendido tanto; a sabedoria é humilde por não saber mais”.  
(William Cowper)

---

# Exercite-se

- ❑ Altere a tabela Produto:
  - ❖ Adicione uma coluna (estoque **integer**).
- ❑ Crie uma função que possa ajustar o valor do estoque de cada produto segundo suas vendas.
  - ❖ A função precisa identificar qual o evento (**insert**, **update** ou **delete**) ocorrido e ajustar o valor do estoque de acordo com a quantidade da venda;
- ❑ Crie um gatilho que seja **AFTER** insert, update ou delete na tabela venda para ajustar o estoque dos produtos vendidos.