

Banco de Dados – IMD0401

Aula 21 – Visão e Gatilho

João Carlos Xavier Júnior

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**.

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

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:

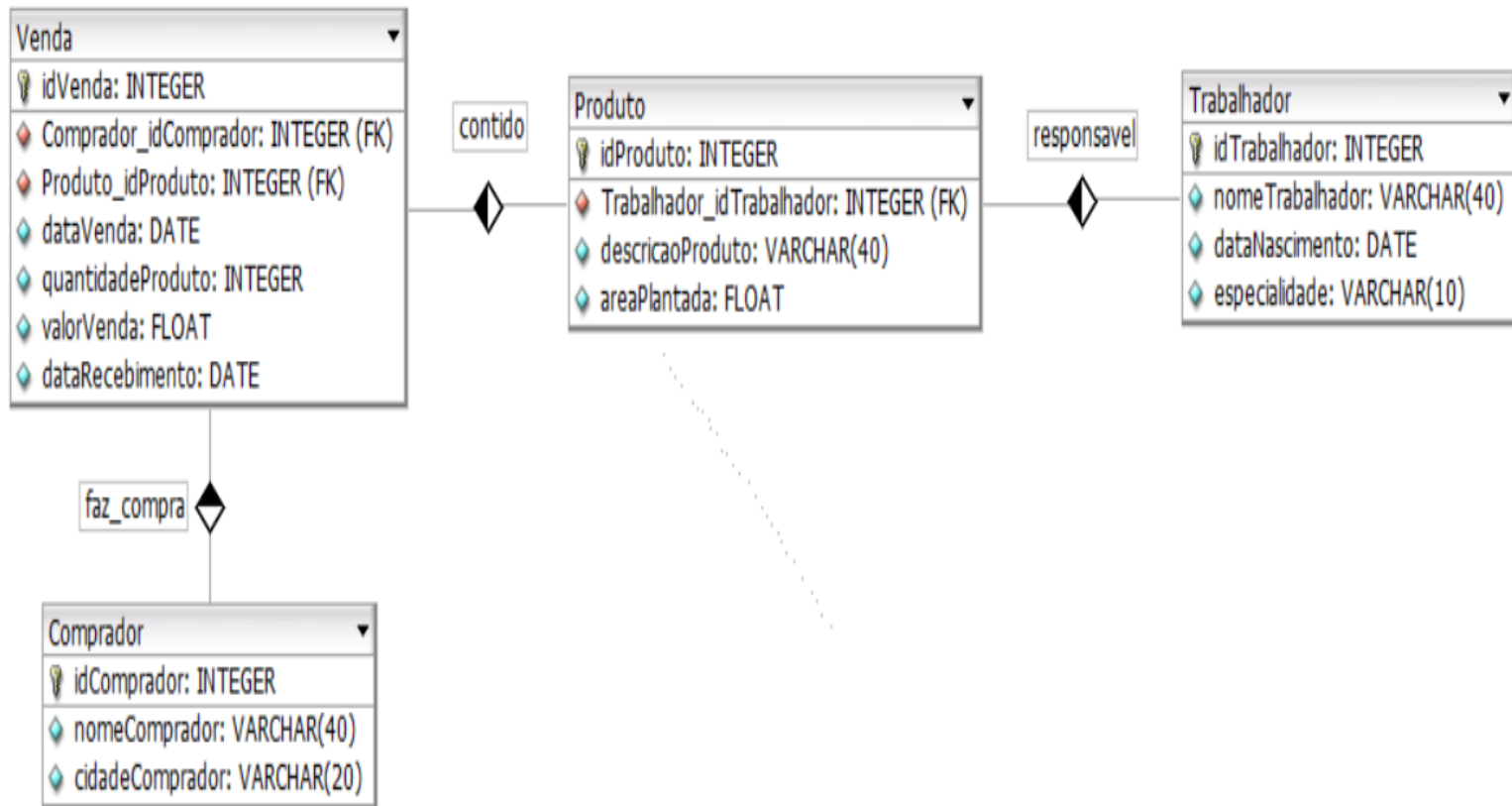
❖ Vantagens:

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



DDL – view

Estudo de caso:



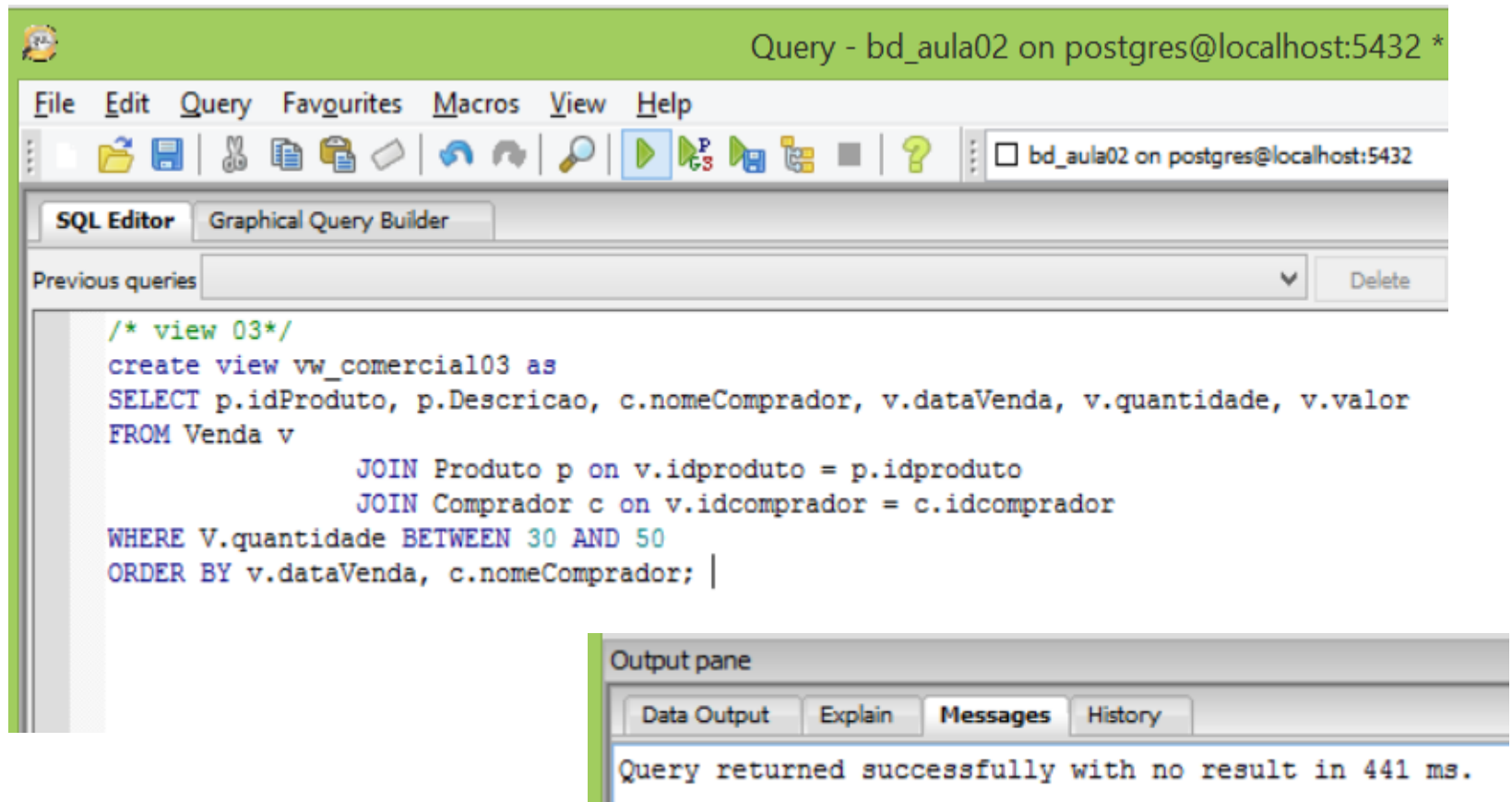
DDL – view

❑ Codificando uma view:

```
/* view 03*/  
create view vw_comercial03 as  
SELECT p.idProduto, p.NomeProduto, c.nomeComprador,  
v.dataVenda, itv.quantidadeProduto, v.valorVenda  
FROM Venda v  
  
JOIN Itens_venda itv on itv.idVenda = v.idVenda  
JOIN Produto p on p.idproduto = itv.idproduto  
JOIN Comprador c on c.idcomprador = v.idcomprador  
  
WHERE itv.quantidadeProduto BETWEEN 30 AND 50  
ORDER BY v.dataVenda, c.nomeComprador;
```

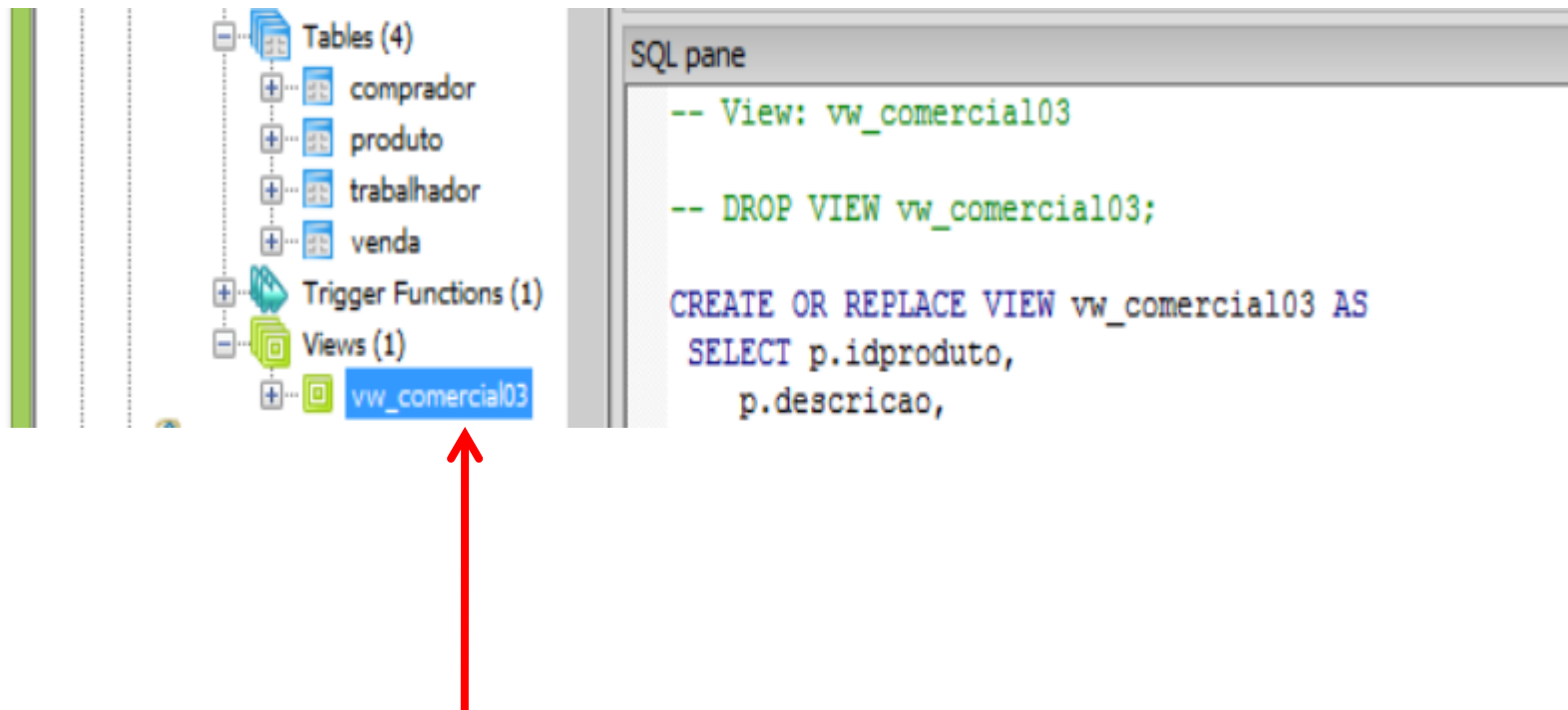
DDL – view

❑ Criando uma view:



DDL – view

❑ Criando uma view:

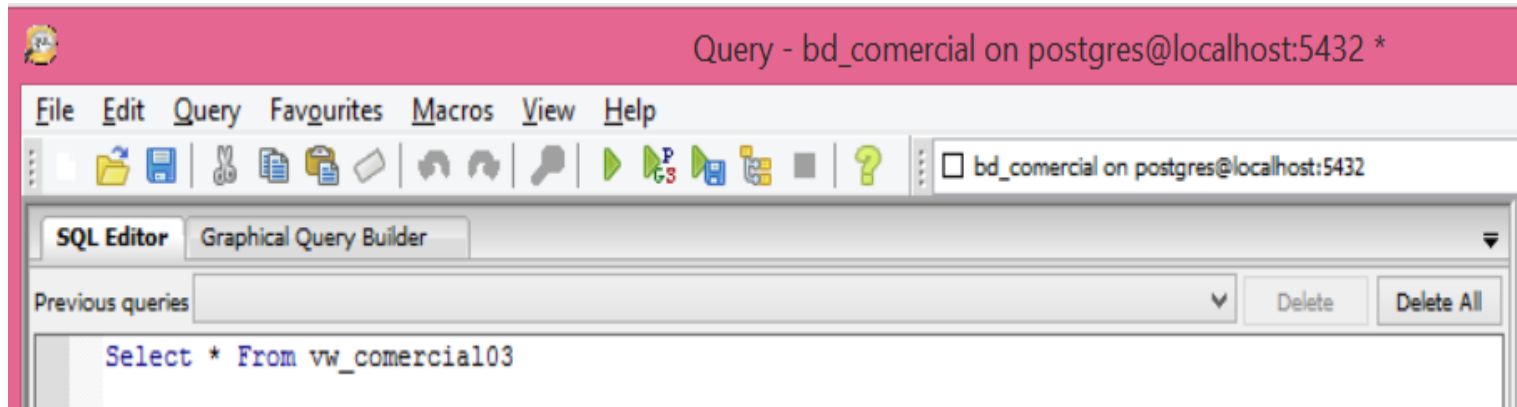


The screenshot displays a database management interface. On the left, a tree view shows the database structure with categories: Tables (4), Trigger Functions (1), and Views (1). Under the Views category, the view 'vw_comercial03' is highlighted in blue. A red arrow points upwards from below the tree view towards the 'vw_comercial03' 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

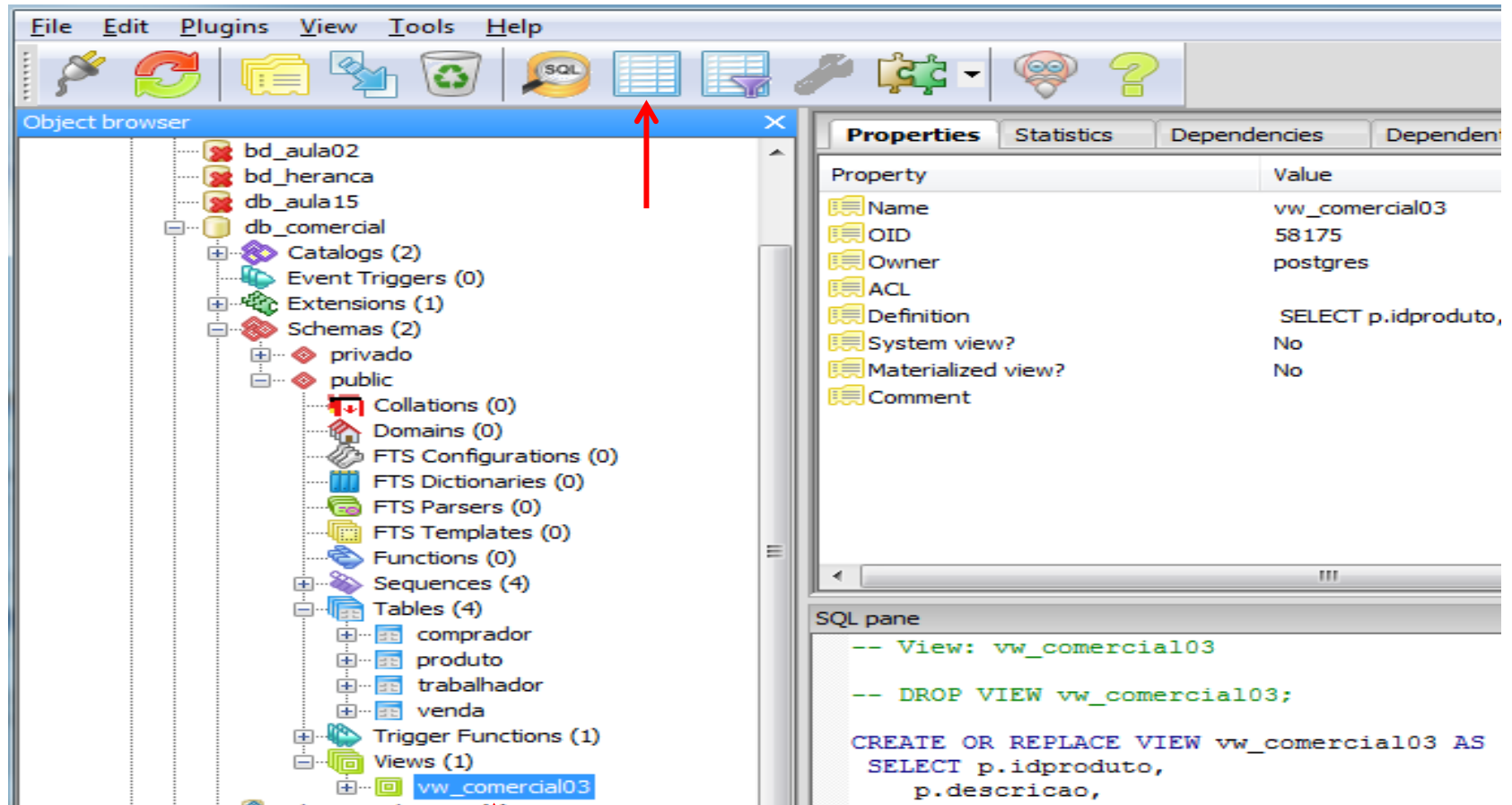
❑ Resultado:

Output pane

	idproduto integer	descricao character varying(50)	nomecomprador character varying(50)	datavenda date	quantidade integer	valor numeric
1	2	Pera	Chico Pira	2016-10-15	35	250
2	3	Laranja	João Pedra	2016-10-26	40	115.45

DDL – view

❑ Executando uma view:



The screenshot displays a database management interface with the following components:

- Object browser:** A tree view on the left showing the database structure. The 'Views' folder is expanded, and 'vw_comercial03' is selected. A red arrow points to this view.
- Properties pane:** A pane on the right showing the properties of the selected view. The 'Definition' property is highlighted, showing the SQL code for the view.
- SQL pane:** A pane at the bottom containing the SQL code to create or replace the view.

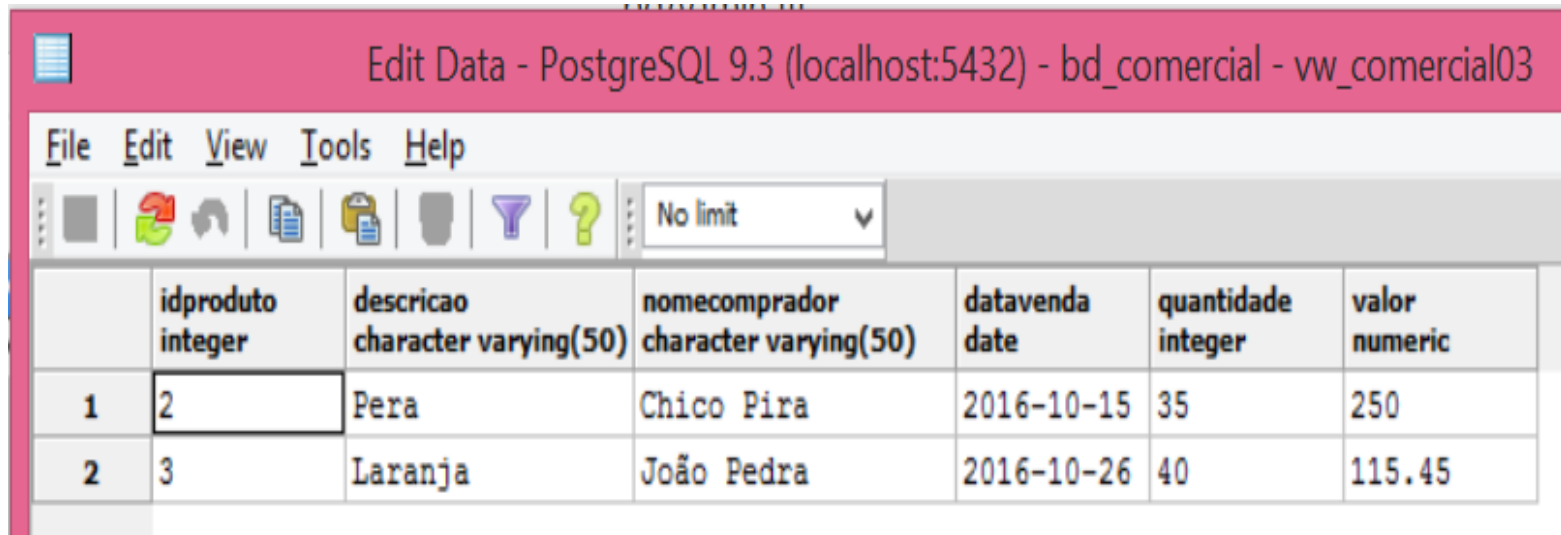
The SQL code in the SQL pane is as follows:

```
-- View: vw_comercial03
-- DROP VIEW vw_comercial03;

CREATE OR REPLACE VIEW vw_comercial03 AS
SELECT p.idproduto,
       p.descricao,
```

DDL – view

❑ Executando uma view:



Edit Data - PostgreSQL 9.3 (localhost:5432) - bd_comercial - vw_comercial03

	idproduto integer	descricao character varying(50)	nomecomprador character varying(50)	datavenda date	quantidade integer	valor numeric
1	2	Pera	Chico Pira	2016-10-15	35	250
2	3	Laranja	João Pedra	2016-10-26	40	115.45

SQL - DQL

❑ Alterando os dados:

Edit Data - PostgreSQL 9.3 (localhost:5432) - bd_comercial - venda

	idvenda [PK] serial	idcomprador integer	idproduto integer	datavenda date	quantidade integer	valor numeric	datarecebimen date
1	8	1	4	2014-08-03	10	150.55	
2	9	2	2	2016-10-15	35	250	
3	10	3	3	2016-10-26	41	115.45	
*							

Edit Data - PostgreSQL 9.3 (localhost:5432) - bd_comercial - venda

	idvenda [PK] serial	idcomprador integer	idproduto integer	datavenda date	quantidade integer	valor numeric	datarecebimen date
1	8	1	4	2014-08-03	10	150.55	
2	9	2	2	2016-10-15	35	250	
3	10	3	3	2016-10-26	41	115.57	
*							

SQL - DQL

❑ Atualizando a view:

Edit Data - PostgreSQL 9.3 (localhost:5432) - bd_comercial - vw_comercial03

	idproduto integer	descricao character vary	nomecomprado character vary	datavenda date	quantidade integer	valor numeric	
1	2	Pera	Chico Pira	2016-10-15	35	250	
2	3	Laranja	João Pedra	2016-10-26	41	115.57	

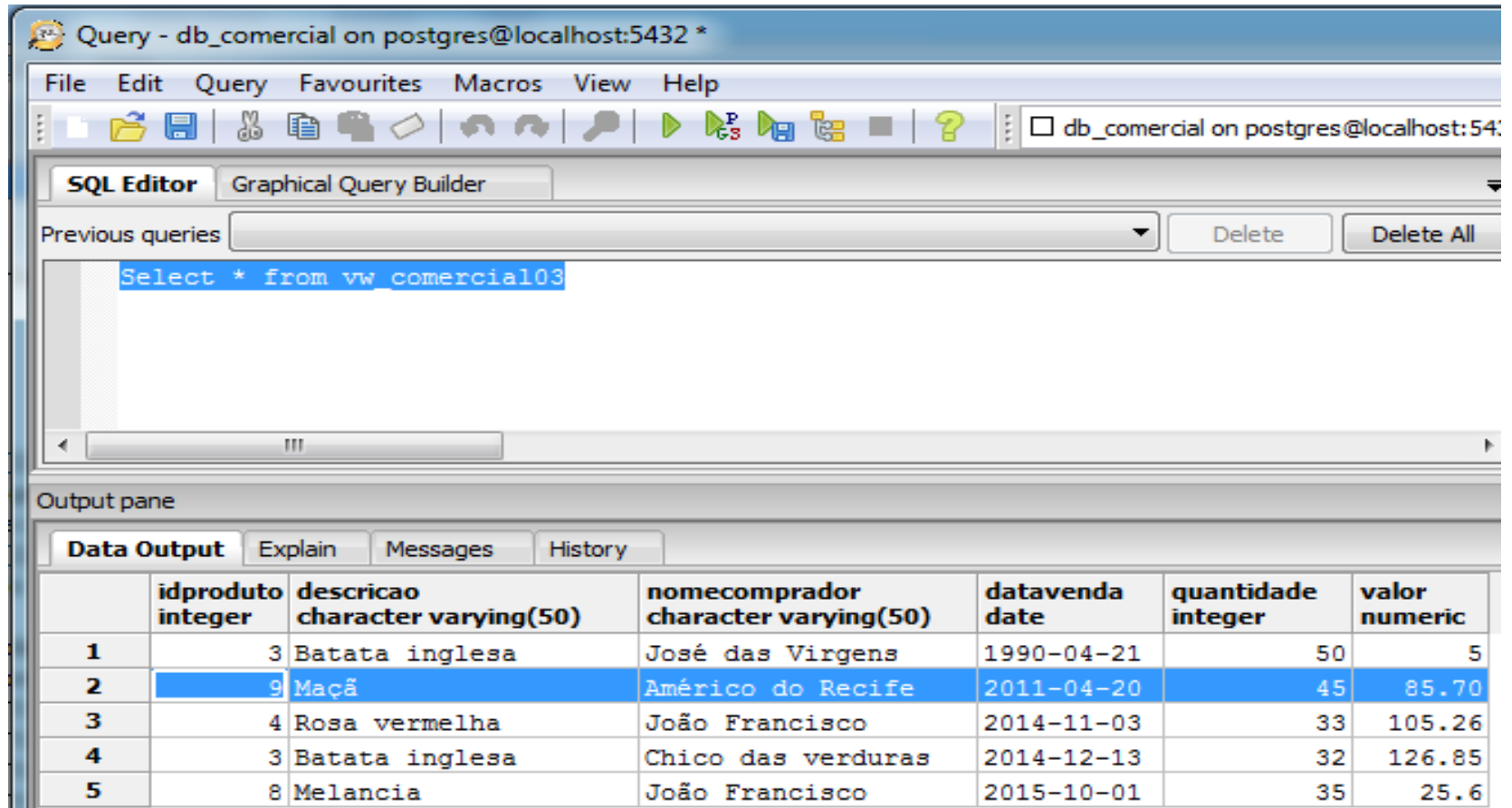
SQL - DQL

❑ 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 contains the query: `Select * from vw comercial03`. Below the editor, there is an "Output pane" with tabs for "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is selected, 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 integer</code>	<code>descricao character varying(50)</code>	<code>nomecomprador character varying(50)</code>	<code>datavenda date</code>	<code>quantidade integer</code>	<code>valor numeric</code>
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

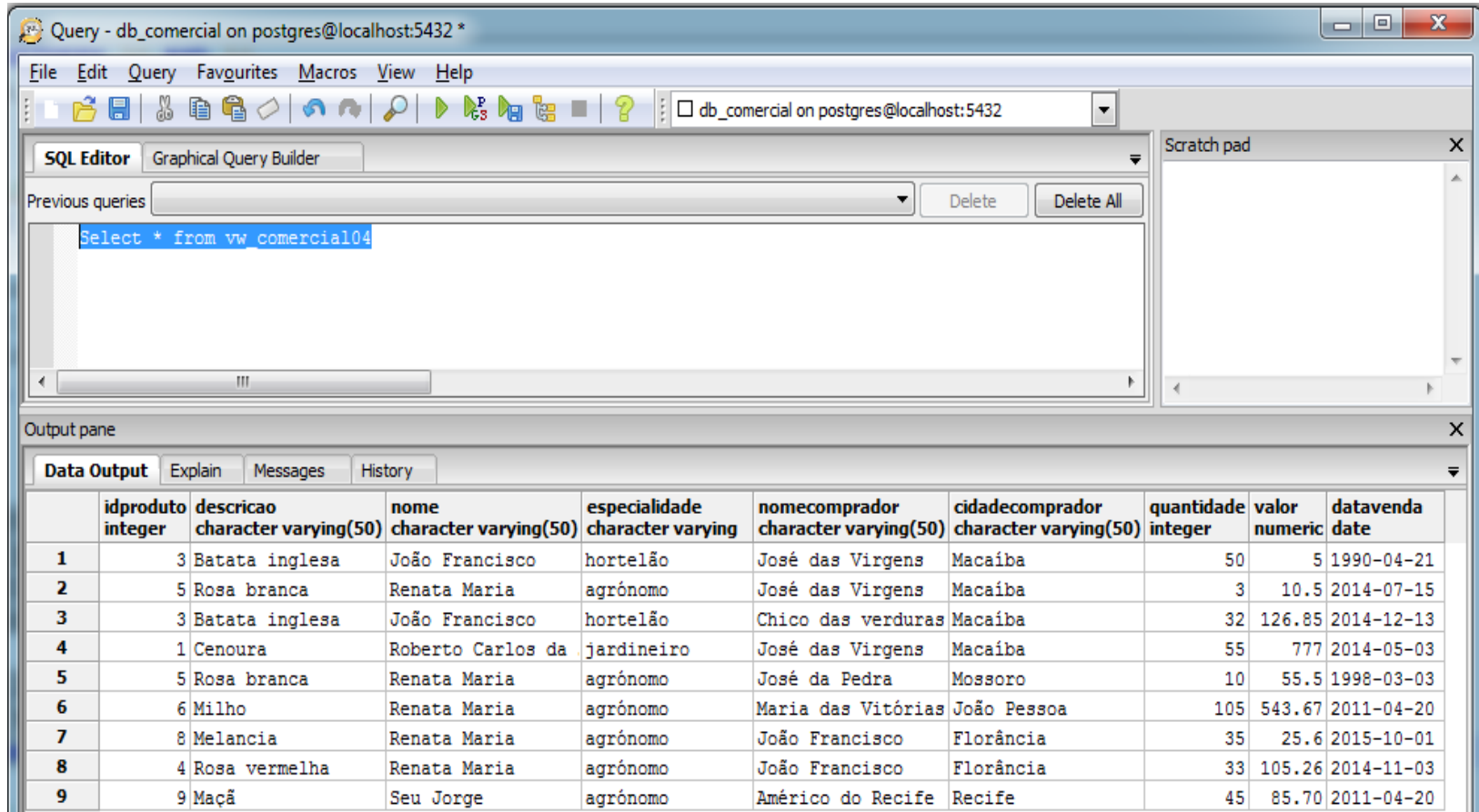
DDL – view

❑ 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:

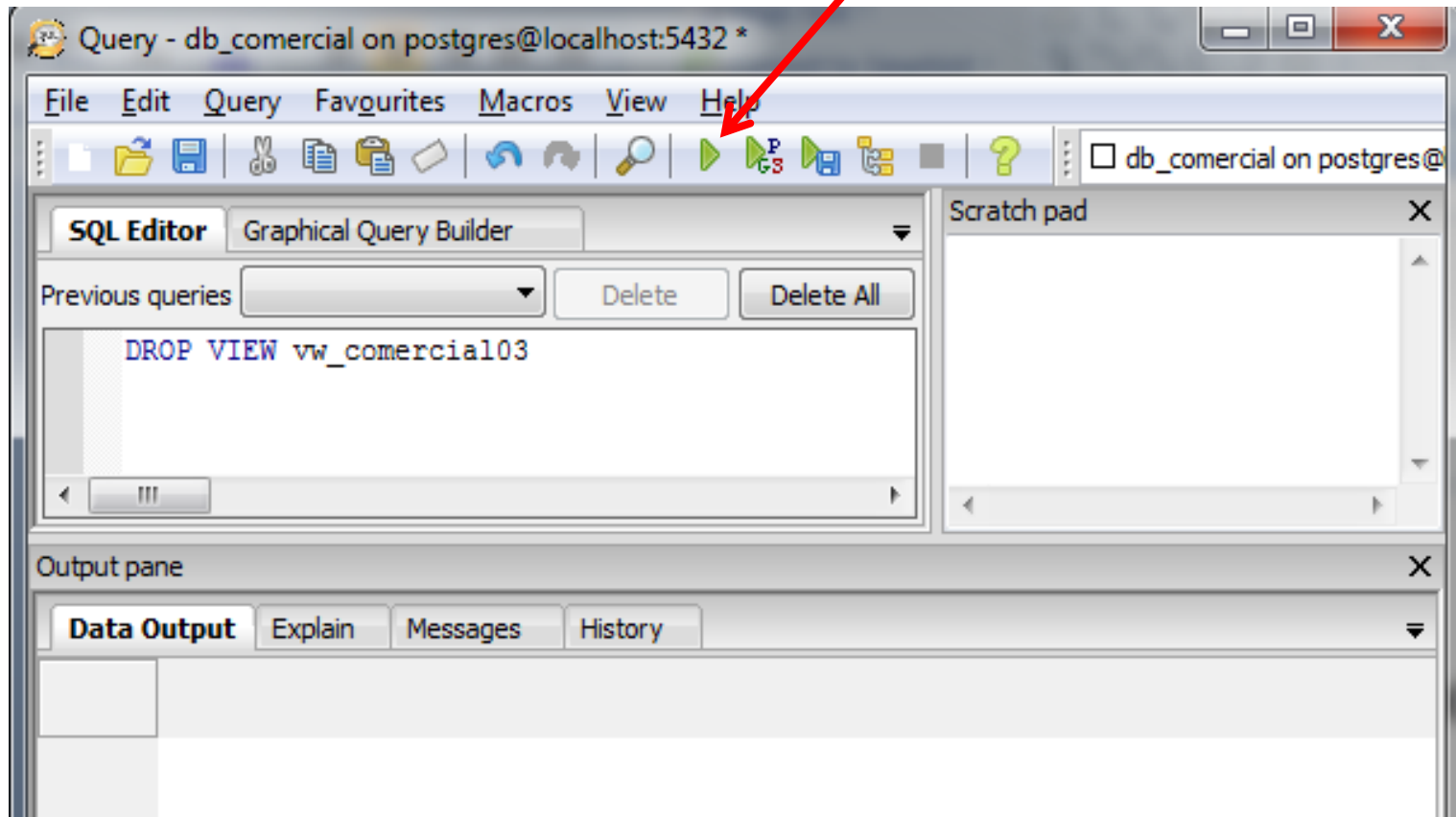


The screenshot shows a PostgreSQL SQL Editor window titled "Query - db_comercial on postgres@localhost:5432 *". The window has a menu bar (File, Edit, Query, Favourites, Macros, View, Help) and a toolbar. The "SQL Editor" tab is active, displaying the query: `Select * from vw_comercial04`. Below the editor is a "Previous queries" list with "Delete" and "Delete All" buttons. To the right is a "Scratch pad" area. At the bottom is the "Output pane" with tabs for "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is selected, showing a table with 9 rows and 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	datavenda 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 – Materialized View

❑ Conceito:

- ❖ View Materializada é uma tabela real no banco de dados.
- ❖ As Views Materializadas melhoram o desempenho em operações de leitura.
- ❖ Os dados das Views materializadas são armazenados em uma tabela.

DDL – Materialized View

❏ Conceito:

- ❖ Ao contrário da View tradicional, que nos apresentam dados atualizados automaticamente, as **Views materializadas** precisam de um mecanismo de atualização.

```
CREATE MATERIALIZED VIEW table_name
[ (column_name [, ...] ) ]
[ WITH ( storage_parameter [= value] [, ...] ) ]
[ TABLESPACE tablespace_name ]
AS query
[ WITH [ NO ] DATA ]
```

<https://www.devmedia.com.br/como-funcionam-as-views-no-postgresql/33808>

DDL – Materialized View










❏ Exemplo:

```
/* Materialized View 01*/  
CREATE MATERIALIZED VIEW mt_vw_comercial as  
SELECT p.idProduto, p.NomeProduto,c.nomeComprador,  
v.dataVenda, itv.quantidadeProduto, v.valorVenda  
FROM Venda v  
  
JOIN Itens_venda itv on itv.idVenda = v.idVenda  
JOIN Produto p on p.idproduto = itv.idproduto  
JOIN Comprador c on c.idcomprador = v.idcomprador  
  
WHERE itv.quantidadeProduto BETWEEN 30 AND 50  
ORDER BY v.dataVenda, c.nomeComprador;
```

DDL – Materialized View

❏ Exemplo:

```
/* Materialized View 01*/  
CREATE MATERIALIZED VIEW mt_vw_comercial as  
SELECT p.idProduto, p.descricaoProduto, c.nomeComprador,  
v.dataVenda, v.quantidadeProduto, v.valorVenda  
FROM Venda v  
  
JOIN Produto p on v.idproduto = p.idproduto  
JOIN Comprador c on v.idcomprador = c.idcomprador  
  
WHERE V.quantidadeProduto BETWEEN 30 AND 50  
ORDER BY v.dataVenda, c.nomeComprador;
```

- ▼  Materialized Views (1)
 - ▼  mt_vw_comercial
 - ▼  Columns (6)
 -  idproduto
 -  descricaoproduto
 -  nomecomprador
 -  datavenda
 -  quantidadeproduto
 -  valorvenda

DDL – Materialized View

❏ Consultando View Materializada:

```
Select * from mt_vw_comercial
```



Data Output							Explain	Messages	Notifications
	<div><div>idproduto</div><div>integer</div></div>	<div><div>descriçãoproduto</div><div>character varying (40)</div></div>	<div><div>nomecomprador</div><div>character varying (40)</div></div>	<div><div>datavenda</div><div>date</div></div>	<div><div>quantidadeproduto</div><div>integer</div></div>	<div><div>valorvenda</div><div>double precision</div></div>			
1	3	Produto 3	José Maria	2020-03-03	50	100.5			
2	3	Produto 3	Teresa Maria	2020-10-30	44	5.1			
3	3	Produto 3	Chico Pira	2021-02-05	32	105.62			

DDL – Materialized View

❏ Consultando View Materializada:








	Data Output	Explain	Messages	Notifications						
	idvenda [PK] integer	idcomprador integer	idproduto integer	datavenda date	quantidadeproduto integer	valorvenda double precision	datarecebimento date			
1	1	7	3	2020-03-03	50	100.5	2020-03-15			
2	2	7	1	2021-02-20	55	56.5	2021-12-13			
3	3	7	5	2020-12-26	3	28.1	[null]			
4	4	3	5	2020-12-15	10	45.6	2021-01-04			
5	5	1	4	2021-01-01	35	15.25	2021-12-13			
6	6	2	3	2021-02-05	32	105.62	[null]			
7	7	1	3	2021-01-15	5	12.5	[null]			
8	8	1	2	2020-09-10	1	10.5	2020-11-25			

DDL – Materialized View

❏ Consultando View Materializada:

```
Select * from mt_vw_comercial
```

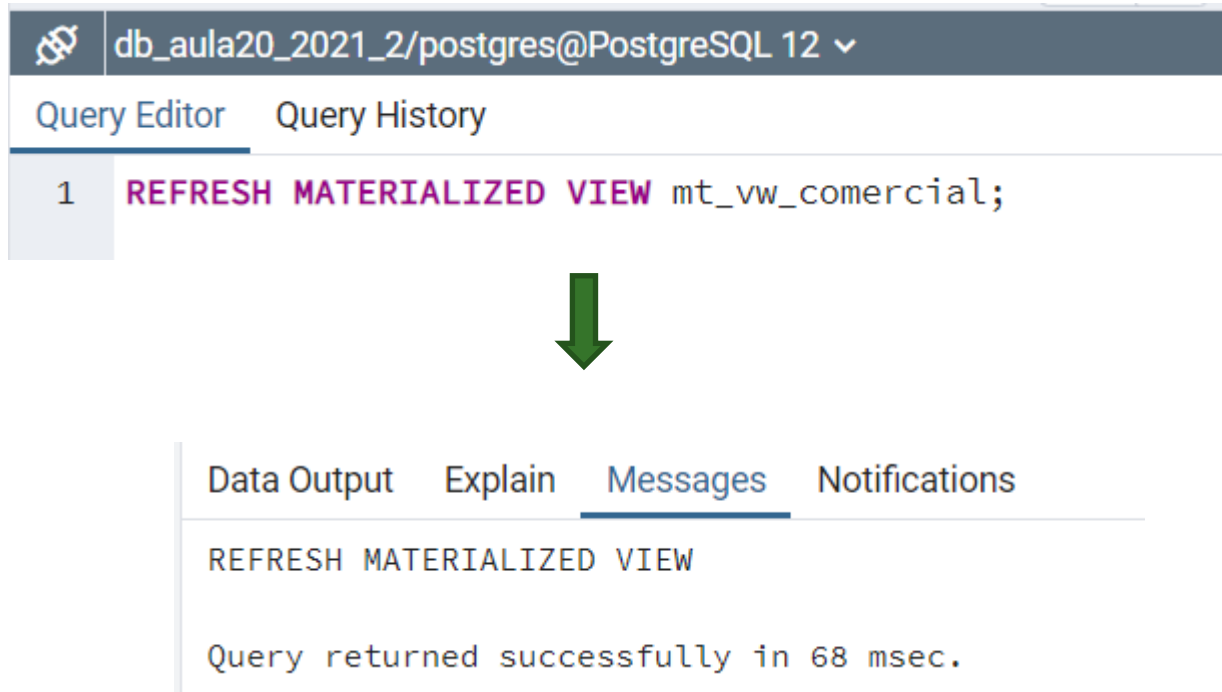


Data Output		Explain	Messages	Notifications		
	 idproduto integer	 descricaoproduto character varying (40)	 nomecomprador character varying (40)	 datavenda date	 quantidadeproduto integer	 valorvenda double precision
1	3	Produto 3	José Maria	2020-03-03	50	100.5
2	3	Produto 3	Teresa Maria	2020-10-30	44	5.1
3	3	Produto 3	Chico Pira	2021-02-05	32	105.62



DDL – Materialized View

❑ Consultando View Materializada:









DDL – Materialized View

❏ Consultando View Materializada:

```
Select * from mt_vw_comercial
```



Data Output		Explain	Messages	Notifications		
	 idproduto integer	 descricaoproduto character varying (40)	 nomecomprador character varying (40)	 datavenda date	 quantidadeproduto integer	 valorvenda double precision
1	3	Produto 3	José Maria	2020-03-03	50	100.5
2	3	Produto 3	Teresa Maria	2020-10-30	44	5.1
3	4	Produto 4	João Maria	2021-01-01	35	15.25
4	3	Produto 3	Chico Pira	2021-02-05	32	105.62

DDL – Materialized View

- ❑ **Quando usar VIEW ou MATERIALIZED VIEW?**
 - ❖ A decisão se a sua view deve ser simples ou materializada é tomada com base no tipo de utilização das tabelas usadas pela consulta da view.
 - ❖ A decisão é simples. Você consulta mais na view do que altera os dados das tabelas?
 - ❖ Os dados do seu banco de dados são alterados com frequência?

DDL – Materialized View

□ Quando usar VIEW ou MATERIALIZED VIEW?

- ❖ Usa-se uma visão materializada quando o desempenho das buscas na view é mais importante que o desempenho da escrita nas tabelas utilizadas por ela.
- ❖ Mas, se uma tabela utilizada pela view tem muita alteração de dados, talvez seja mais interessante que a view não seja materializada.

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:

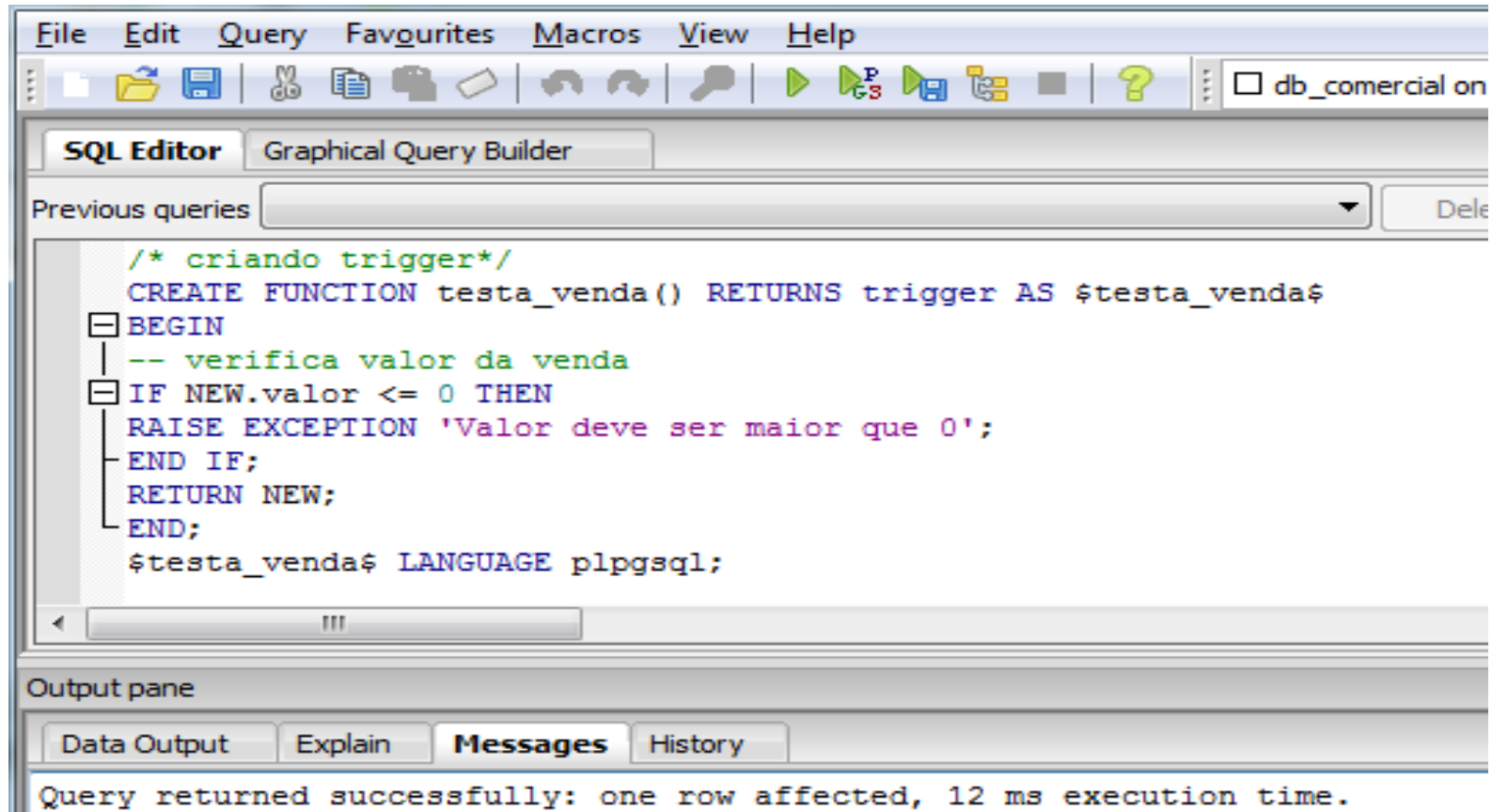
```
CREATE TRIGGER nome { BEFORE | AFTER } {  
evento [ OR ... ] }  
ON tabela [ FOR [ EACH ] { ROW | STATEMENT  
} ]  
EXECUTE PROCEDURE nome_da_função (  
argumentos )
```

❑ Observação:

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

DDL – Trigger

❑ Criando e especificando uma outra função:



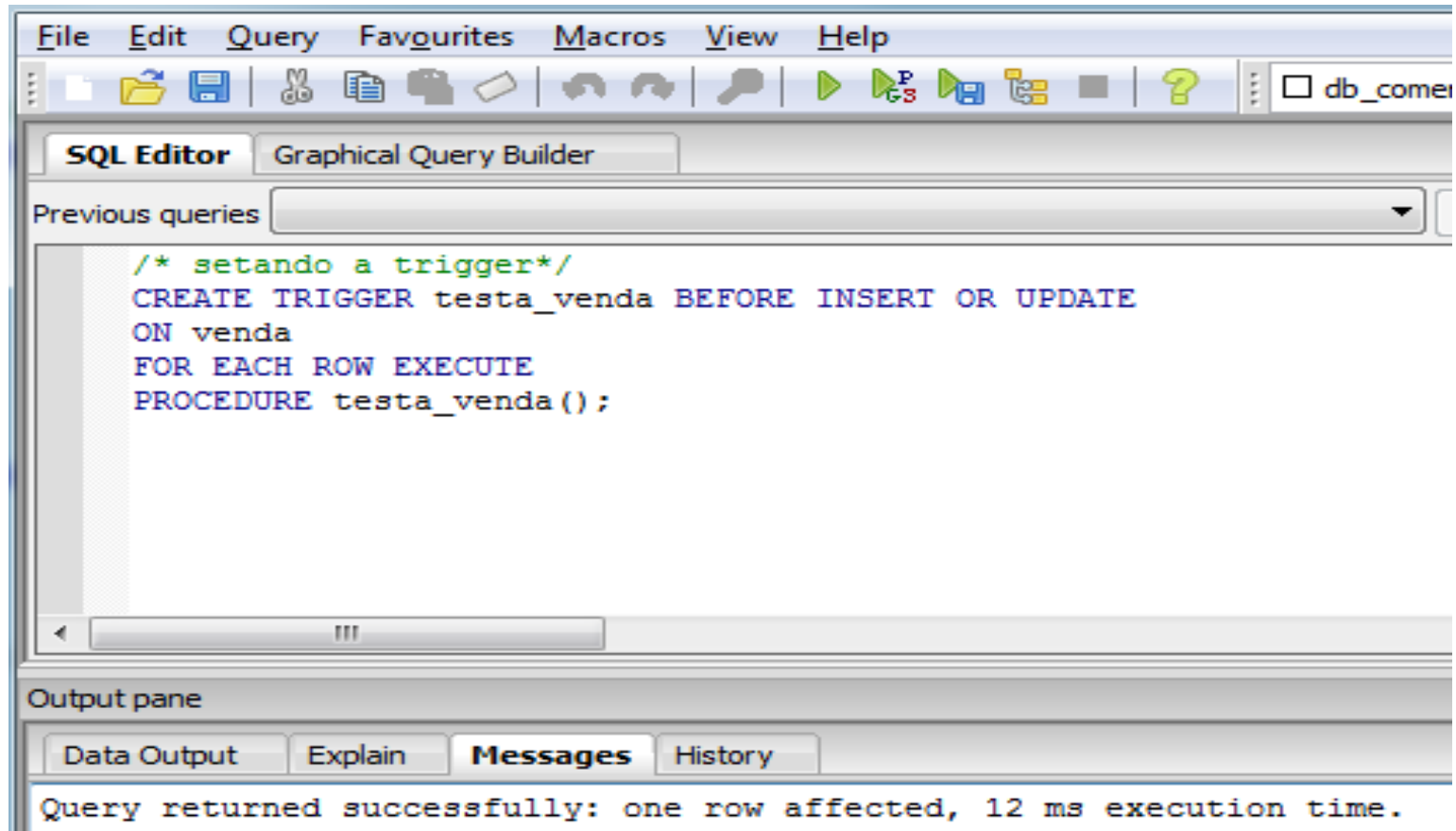
The screenshot shows an SQL Editor window with a menu bar (File, Edit, Query, Favouirites, Macros, View, Help) and a toolbar. The main text area contains the following SQL code:

```
/* 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, showing the message: "Query returned successfully: one row affected, 12 ms execution time."

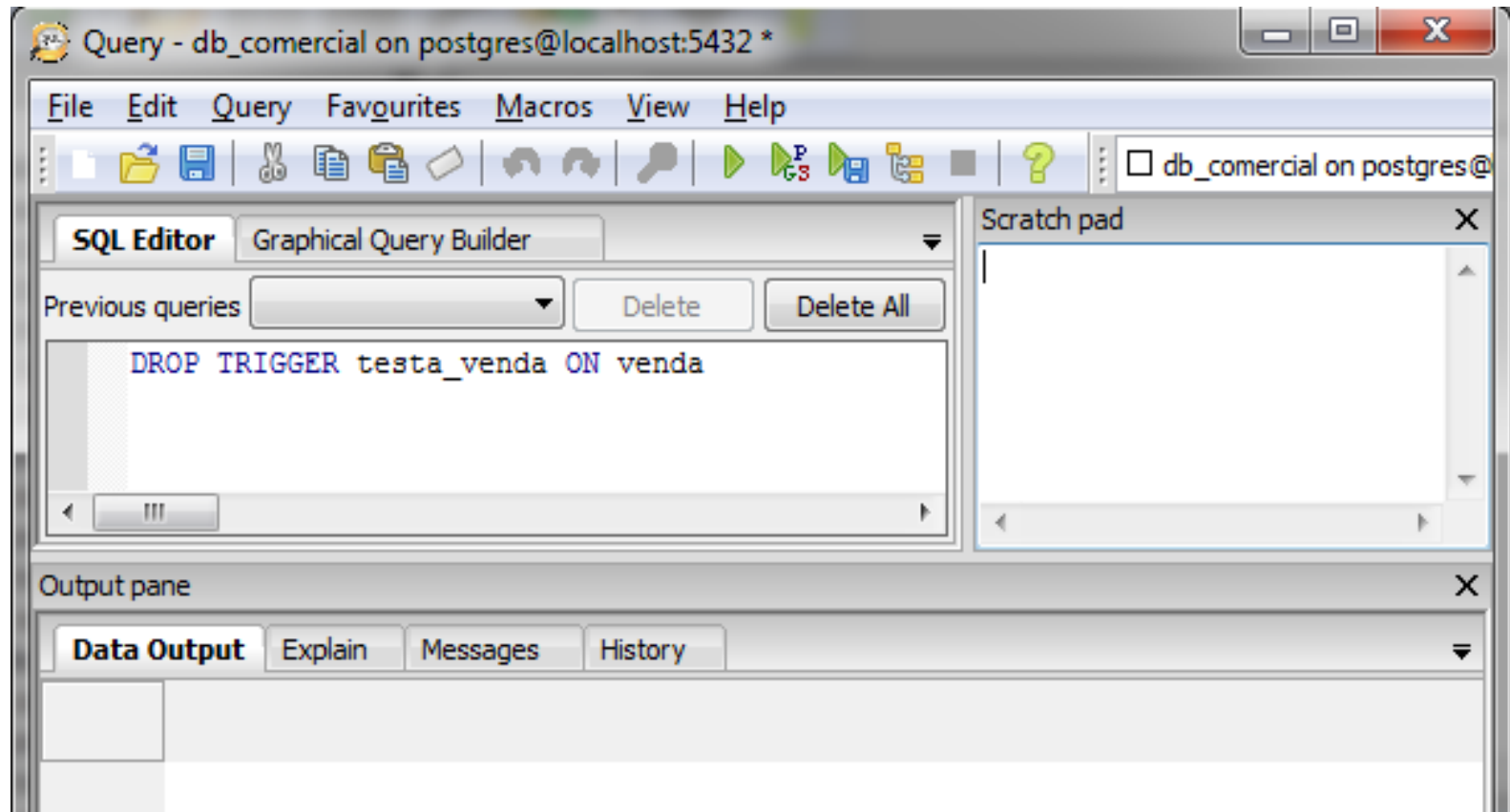
DDL – Trigger

- ❑ Criando e especificando um outro gatilho:



DDL – Trigger

❑ Dropando (DROP) um Gatilho:



Dúvidas...



Atividade

- ❑ 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.

Solução

```
/* Alterar produto*/  
ALTER TABLE produto ADD estoque integer;  
  
/* Atualizar as quantidades de estoque*/  
UPDATE produto set estoque = 10 where idproduto > 0;
```

Solução

```
CREATE FUNCTION f_controlar_estoque() RETURNS TRIGGER AS
$t_controla_estoque$
BEGIN
    IF (TG_OP = 'DELETE') THEN
        Update produto
        Set estoque = estoque + old.quantidadeProduto
        where idproduto = old.idproduto;
    ELSIF (TG_OP = 'UPDATE') THEN
        Update produto
        Set estoque = estoque - (new.quantidadeProduto -
old.quantidadeProduto)
        where idproduto = old.idproduto;
    ELSIF (TG_OP = 'INSERT') THEN
        Update produto
        Set estoque = estoque - new.quantidadeProduto
        where idproduto = new.idproduto;
    END IF;
    RETURN NULL; -- result is ignored since this is an AFTER trigger
END;
$t_controla_estoque$ LANGUAGE plpgsql;

CREATE TRIGGER t_controla_estoque
AFTER INSERT OR UPDATE OR DELETE ON Itens_Venda
FOR EACH ROW EXECUTE PROCEDURE f_controlar_estoque();
```