

Banco de Dados II

SQL Avançado

Dr^a Alana Moraes

Objetivo da Aula de hoje

Começar com SQL Avançado

Acabamos com essa parte de comandos básicos

GROUP BY, HAVING

JOIN

Base de dados inicial da Aula

Considerando a base de dados da última aula

Caso não visualize a base, crie o um banco de dados e rode o `compras_completo.sql` pelo pgadmin.

Base relacionada ao registro de compras do cartão de crédito.

Vocês revisaram os comandos?

MAX? MIN?

- Qual a compra mais cara realizada nos registros?

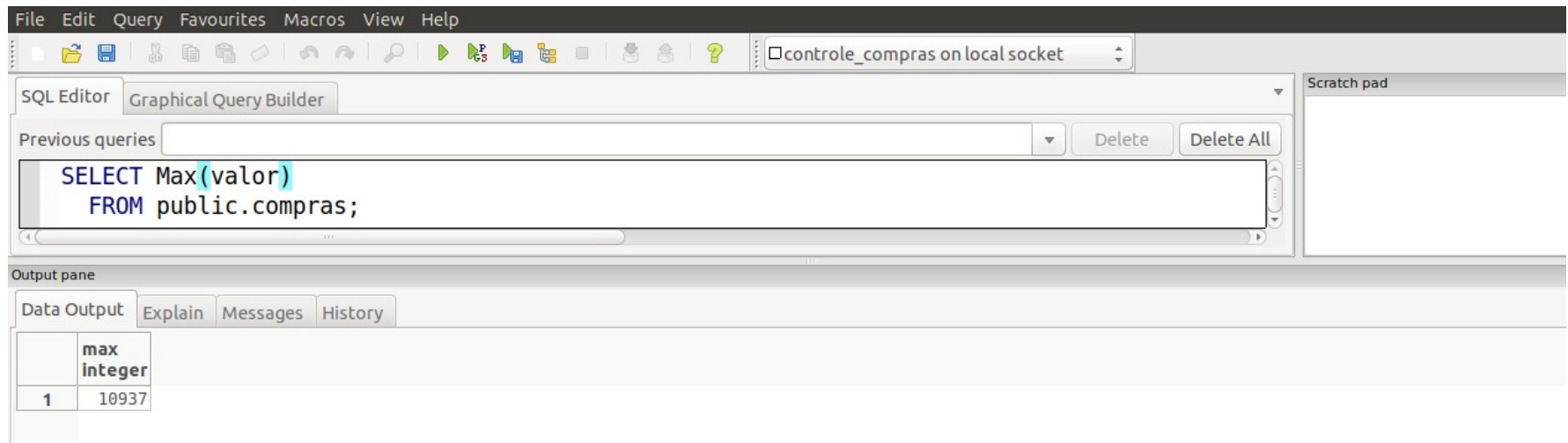
SUM?

- Qual o somatório de todos os valores relacionados à compra LANCHONETE?

Vocês revisaram os comandos?

MAX?

- Qual a compra mais cara realizada nos registros?



The screenshot shows a database management interface with a menu bar (File, Edit, Query, Favourites, Macros, View, Help) and a toolbar. The main window is divided into several panes. The 'SQL Editor' pane contains the following SQL query:

```
SELECT Max(valor)
FROM public.compras;
```

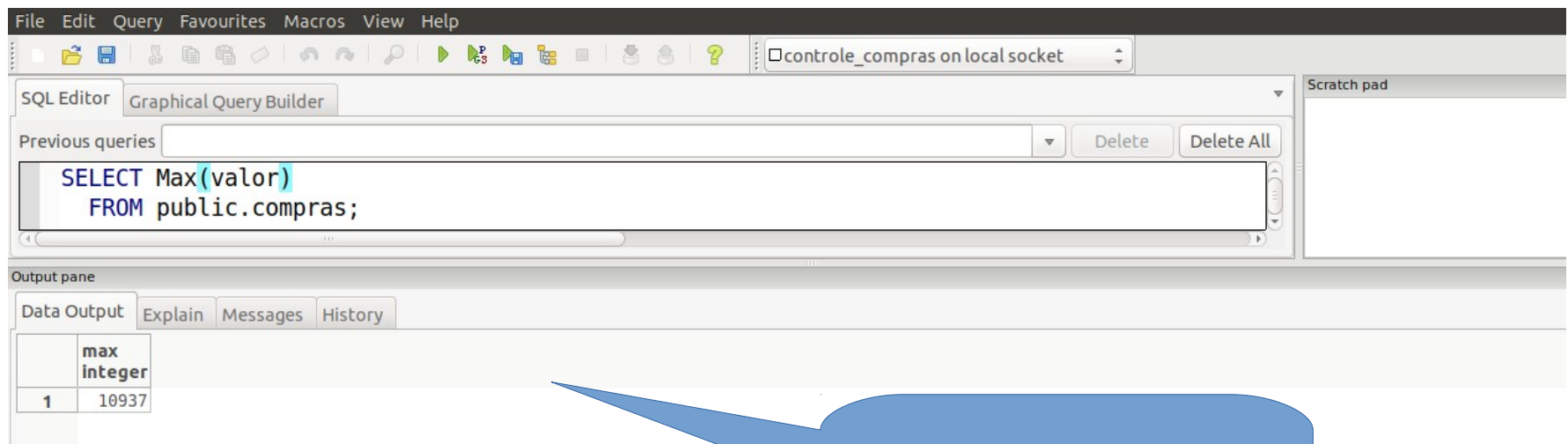
The 'Output pane' is active, showing the results of the query in a table format. The table has two columns: an index and the maximum value.

	max integer
1	10937

Vocês revisaram os comandos?

MAX?

- Qual a compra mais cara realizada nos registros?



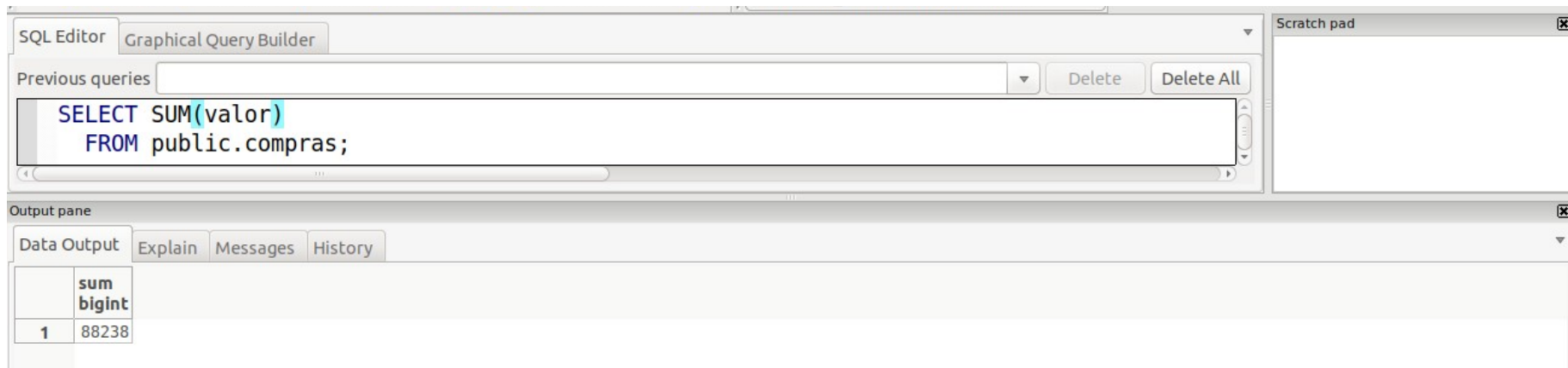
The screenshot shows a database query tool interface. The SQL Editor contains the query: `SELECT Max(valor)
FROM public.compras;`. The Output pane shows the result of the query in a table format:

	max integer
1	10937

A blue speech bubble points to the result table.

E a mais barata?

Vocês revisaram os comandos?



SUM?

- Qual o somatório de todos os valores relacionados à compra LANCHONETE?

**E se eu quiser extrair a soma das
compras por observação?**

Como eu faço isso?

E se eu quiser extrair a soma das compras por observação?

Como eu faço isso?

GROUP BY

aggregate_function: SUM, AVG, COUNT, etc.

```
1 SELECT
2   column_1,
3   aggregate_function (column_2)
4 FROM
5   tbl_name
6 GROUP BY
7   column_1
```

E se eu quiser extrair a soma das compras por observação?

The screenshot shows a database management interface with a top toolbar, a central SQL Editor, a right-side Scratch pad, and a bottom Output pane. The SQL Editor contains a query to find the maximum value for each observation. The Output pane displays the results in a table format.

SQL Editor: `SELECT MAX(valor), observacao
FROM public.compras
GROUP BY observacao;`

Output pane: Data Output

	max integer	observacao character varying(255)
1	2498	COMPRAS DE JANEIRO
2	2000	FIM DE ANO EM PARIS
3	4780	SALA DE ESTAR
4	200	MATERIAL ESCOLAR
5	1709	PARCELA DA CASA
6	909	IPVA
7	576	SAPATOS
8	10937	CARNAVAL EM CUNCUN
9	828	FESTA
10	1203	QUARTOS
11	632	IPTU
12	403	COPA
13	1501	PRESENTE DA SOGRA
14	954	SHOW DA IVETE SANGALO
15	678	PASSAGEM PRA BAHIA

OK. Unix Ln 3, Col 22, Ch 74 27 rows. 13 msec

**E se eu quiser extrair a soma das compras
por observação?**

Agora quero compras acima de 100,00

E se eu quiser extrair a soma das compras por observação?

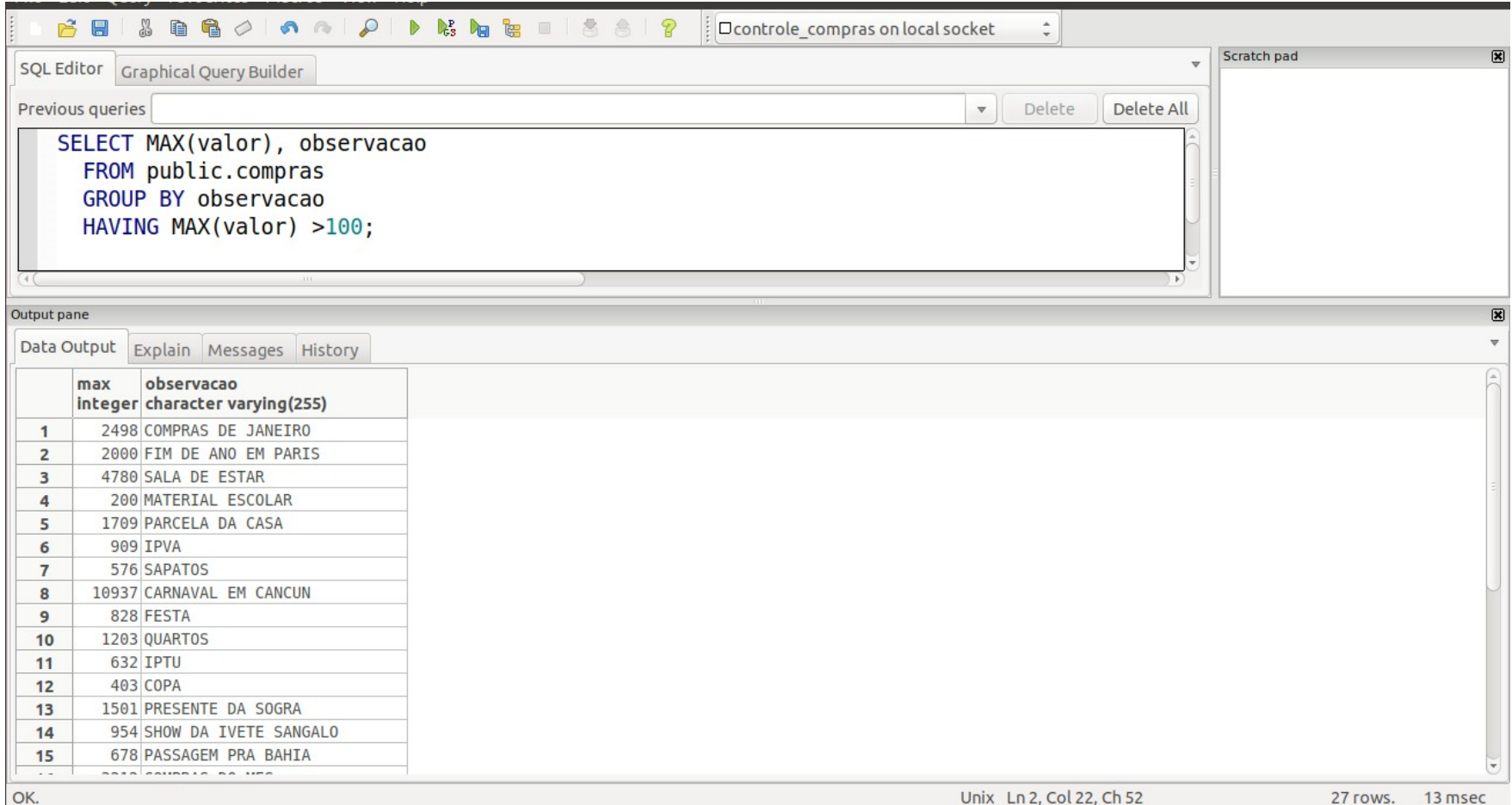
Agora quero compras acima de 100,00

Comando:

HAVING

```
1 SELECT
2   column_1,
3   aggregate_function (column_2)
4 FROM
5   tbl_name
6 GROUP BY
7   column_1
8 HAVING
9   condition;
```

E se eu quiser extrair a soma das compras por observação?



The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, search, and execution. The main window is divided into three panes: SQL Editor, Output pane, and Scratch pad.

SQL Editor: Contains the following SQL query:

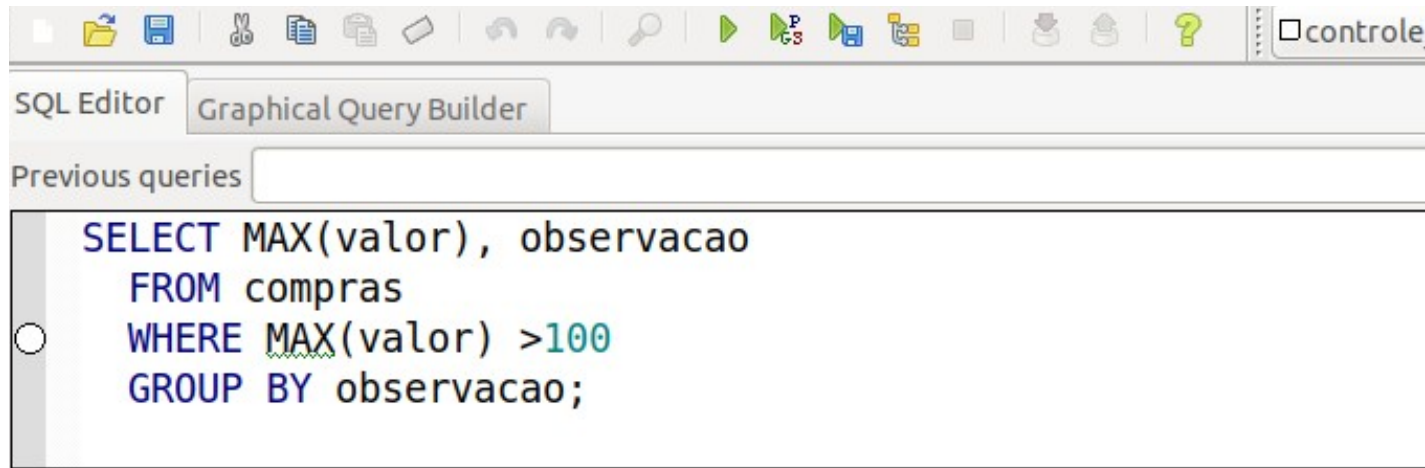
```
SELECT MAX(valor), observacao
FROM public.compras
GROUP BY observacao
HAVING MAX(valor) >100;
```

Output pane: Displays the results of the query in a table format. The table has two columns: **max integer** and **observacao character varying(255)**. The results are as follows:

	max integer	observacao character varying(255)
1	2498	COMPRAS DE JANEIRO
2	2000	FIM DE ANO EM PARIS
3	4780	SALA DE ESTAR
4	200	MATERIAL ESCOLAR
5	1709	PARCELA DA CASA
6	909	IPVA
7	576	SAPATOS
8	10937	CARNAVAL EM CANCUN
9	828	FESTA
10	1203	QUARTOS
11	632	IPTU
12	403	COPA
13	1501	PRESENTE DA SOGRA
14	954	SHOW DA IVETE SANGALO
15	678	PASSAGEM PRA BAHIA

The bottom status bar indicates the execution was successful with the message "OK.", the current cursor position is "Unix Ln 2, Col 22, Ch 52", and the query executed in "27 rows. 13 msec".

Eu consigo fazer isso com o Where?



The screenshot shows a SQL Editor window with a toolbar at the top containing icons for file operations, execution, and help. Below the toolbar are tabs for 'SQL Editor' and 'Graphical Query Builder'. A 'Previous queries' list is empty. The main text area contains the following SQL query:

```
SELECT MAX(valor), observacao
FROM compras
WHERE MAX(valor) > 100
GROUP BY observacao;
```

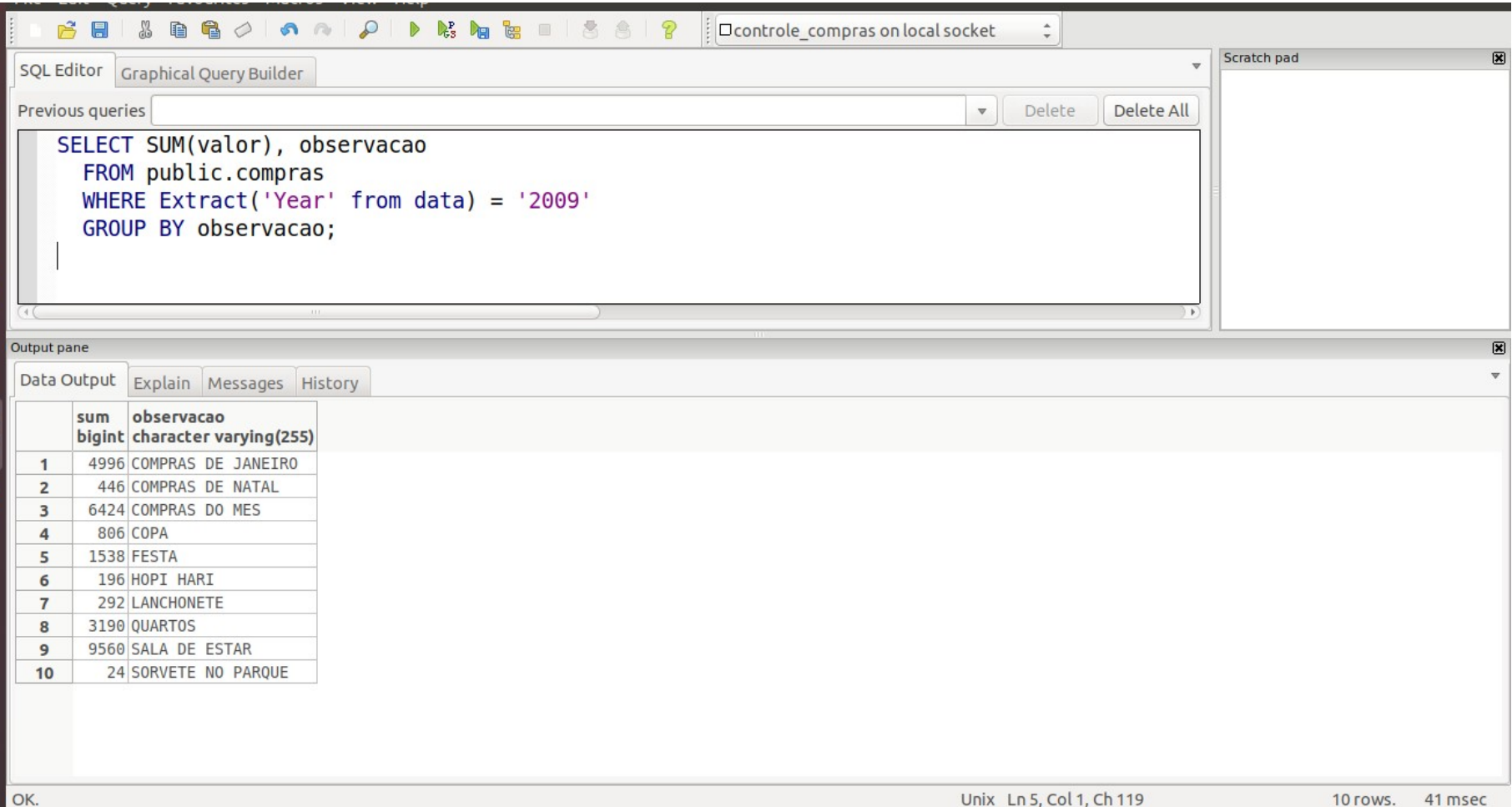
O que vocês acham
que acontecerá?

Where x Having

- A cláusula HAVING define a condição para as linhas do grupo criadas pela cláusula GROUP BY após a aplicação da cláusula GROUP BY
- Enquanto a cláusula WHERE define a condição para linhas individuais antes da cláusula GROUP BY se aplicar.

**E se eu quiser extrair a soma das
compras por observação de 2009?**

E se eu quisesse extrair a soma das compras por observação de 2009?



The screenshot shows a SQL IDE interface with a query editor and an output pane. The query editor contains the following SQL query:

```
SELECT SUM(valor), observacao
FROM public.compras
WHERE Extract('Year' from data) = '2009'
GROUP BY observacao;
```

The output pane displays the results of the query in a table format. The table has two columns: **sum bigint** and **observacao character varying(255)**. The results are as follows:

	sum bigint	observacao character varying(255)
1	4996	COMPRAS DE JANEIRO
2	446	COMPRAS DE NATAL
3	6424	COMPRAS DO MES
4	806	COPA
5	1538	FESTA
6	196	HOPI HARI
7	292	LANCHONETE
8	3190	QUARTOS
9	9560	SALA DE ESTAR
10	24	SORVETE NO PARQUE

The status bar at the bottom indicates: OK. Unix Ln 5, Col 1, Ch 119 10 rows. 41 msec

Sintaxe Select

Select [distinct] {*,colunas [alias], expressões,
funções}

From {tabelas [alias]}

[Where condições]

[Group by colunas]

[Having condição]

[Order by colunas [ASC|DESC]];

Um pouco de Álgebra Relacional

Seleção

$$\sigma_F (R)$$

Projeção

$$\pi_{i_1, i_2, \dots, i_m}(R)$$

Produto Cartesiano

$$R \times S$$

Junção

$$R \bowtie S$$

União

$$R \cup S$$

Interseção

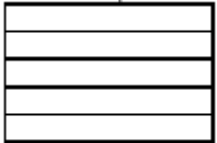
$$R \cap S$$

Diferença

$$R - S$$

Visualizando as operações

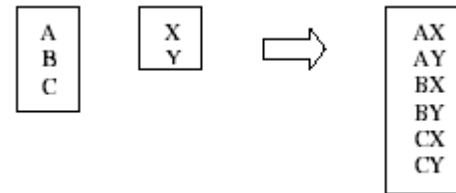
Seleção



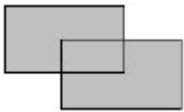
Projeção



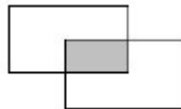
Produto Cartesiano



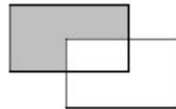
União



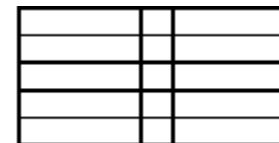
Interseção



Diferença



Junção



Exemplo: Álgebra Relacional e SQL

$\pi_{\text{nome}}(\sigma_{\text{cidade} = \text{'João Pessoa'}}(\text{cliente}))$

=

Select nome

From cliente

Where cidade = 'João Pessoa';

Exemplo: Álgebra Relacional e SQL

- Consulta: Quais clientes têm prazo de entrega maior que 15 dias e são da Paraíba ou Pernambuco?
- Solução:

$\pi_{(\text{nome_cli}, \text{UF})} (\sigma_{((\text{uf} = \text{'PB'} \vee \text{uf} = \text{'PE'}) \wedge (\text{prazoentrega} > 15))}$
 $(\text{cliente} \bowtie \text{Pedido}))$

Select nome, UF

from cliente JOIN pedido on cliente.codcli =
pedido.codcli

where UF in ('PB','PE') and prazoentrega > 15;

Vamos melhorar nossas consultas?

Crie o banco bancoJoins

Rode o script bancoJoins.sql

Exemplo



Basket A



Basket B

Como são nossos consultas atualmente

Realizadas, na maioria dos casos, em uma única tabela.

Muitas vezes, será necessário selecionar dados de duas ou mais tabelas.

JOIN (junções)

Tipos de JOINS

Existem as Junções com estilo non-ANSI ou theta (junção com WHERE).

Junções ANSI join (com JOIN).

As junções ANSI podem ser de dois tipos, as INNER JOINS e as OUTER JOINS. A padrão é a INNER JOIN.

INNER JOIN pode ser escrito com apenas JOIN.

Junções (JOINS)

Junção de duas tabelas (reais ou derivadas) de acordo com as regras do tipo particular de JOIN:

INNER JOIN A tabela resultado contém todos os resultados de ambas as tabelas que satisfaçam a condição fornecida;

OUTER JOIN A tabela resultado contém todas os resultados de ambas as tabelas, ainda que a condição não seja satisfeita;

CROSS JOIN Produto cartesiano dos campos das duas tabelas.

Junções (JOINS)

CROSS JOIN

- ✓ Produto Cartesiano
- ✓ Permite que você produza o produto cartesiano de linhas em duas ou mais tabelas.
- ✓ Diferente dos outros operadores JOIN, como LEFT JOIN ou INNER JOIN, o CROSS JOIN não possui nenhuma condição correspondente na cláusula join.

Junções (JOINS)

CROSS JOIN

```
SELECT basket_a.id,  
       basket_a.fruit,  
       basket_b.id,  
       basket_b.fruit  
FROM basket_a  
CROSS JOIN basket_b;
```

Junções (JOINS)

CROSS JOIN

SELECT basket_a.id,

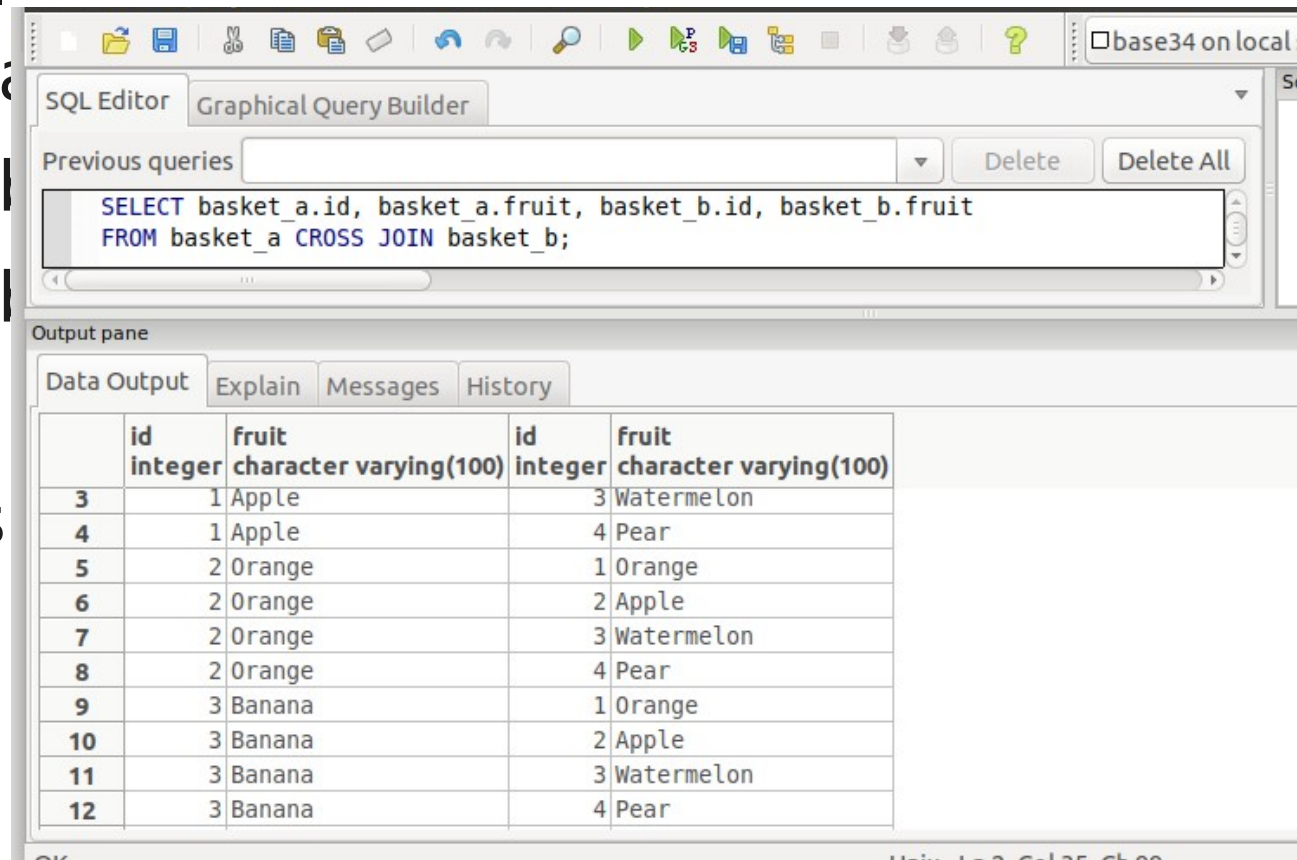
basket_a.fruit,

basket_b.id,

basket_b.fruit

FROM basket_a

CROSS JOIN basket_b



The screenshot shows a SQL IDE interface with a toolbar at the top. The 'SQL Editor' tab is active, displaying the following query:

```
SELECT basket_a.id, basket_a.fruit, basket_b.id, basket_b.fruit
FROM basket_a CROSS JOIN basket_b;
```

Below the editor is the 'Output pane' with tabs for 'Data Output', 'Explain', 'Messages', and 'History'. The 'Data Output' tab is selected, showing a table with 12 rows of results from the CROSS JOIN.

	id integer	fruit character varying(100)	id integer	fruit character varying(100)
3	1	Apple	3	Watermelon
4	1	Apple	4	Pear
5	2	Orange	1	Orange
6	2	Orange	2	Apple
7	2	Orange	3	Watermelon
8	2	Orange	4	Pear
9	3	Banana	1	Orange
10	3	Banana	2	Apple
11	3	Banana	3	Watermelon
12	3	Banana	4	Pear

Junções (JOINS)

INNER JOIN

Normalmente, consiste em juntar duas ou mais tabelas, ligando-as por meio da Chave Primária de uma e a Chave Estrangeira da outra tabela (Mas não se restringe a isto).

São apresentados os registros em que exista ligação entre as tabelas

Existe uma “qualificação”:

Produto cartesiano + qualificação

Equação de Junção:

Tabela1.campoPK = tabela2.campoFK

Junções (JOINS)

Inner Join

```
SELECT cliente.codcli, nome,  
        numped, pedido.codcli  
FROM cliente, pedido  
WHERE cliente.codcli = pedido.codcli;  
ou  
SELECT cliente.codcli, nome,  
        numped, pedido.codcli  
FROM cliente  
JOIN pedido ON cliente.codcli = pedido.codcli;
```


Qualificação, apelidos/alias

Qualificadores de Nome:

- Produto.descricao
- Artista.nome

Operação renomear: Produto p, Itens I

- P.descricao
- $P.codprod = I.codprod$

Inner JOIN com renomeação

```
SELECT nome, p.numped  
FROM cliente c  
INNER JOIN pedido p  
      ON c.codcli = p.codcli;
```

Qual seriam as saídas no exemplos das cestas?

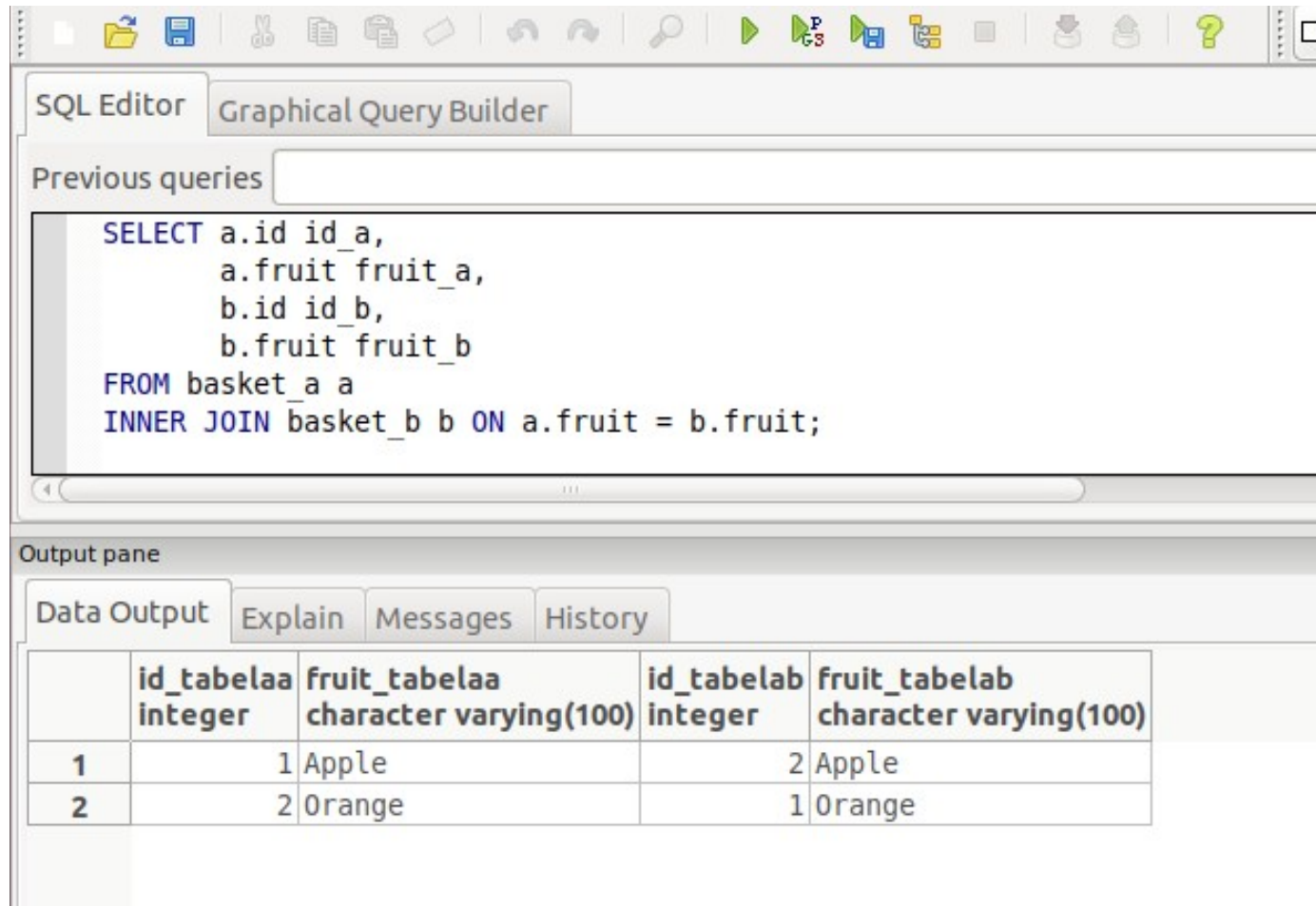
Junções (JOINS)

INNER JOIN

```
SELECT    a.id id_a,  
          a.fruit fruit_a,  
          b.id id_b,  
          b.fruit fruit_b  
FROM basket_a a  
INNER JOIN basket_b b ON a.fruit = b.fruit;
```

Junções (JOINS)

INNER JOIN



The screenshot shows a database application interface with a toolbar at the top. Below the toolbar, there are two tabs: "SQL Editor" and "Graphical Query Builder". The "SQL Editor" tab is active, displaying the following SQL query:

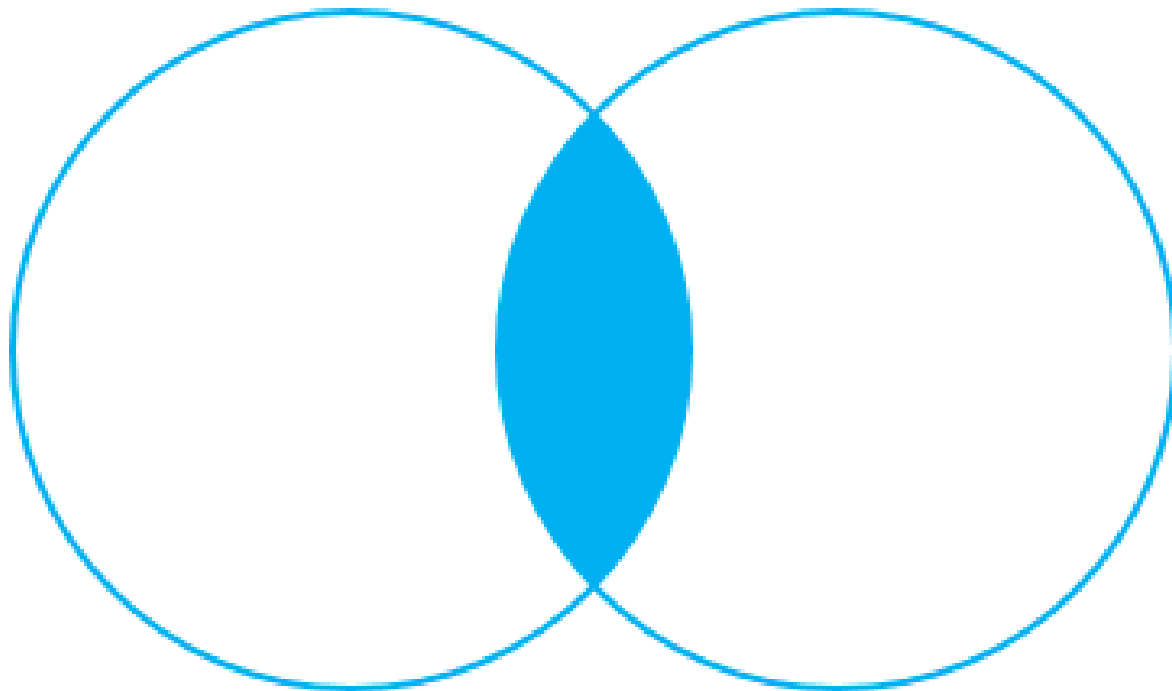
```
SELECT a.id id_a,  
       a.fruit fruit_a,  
       b.id id_b,  
       b.fruit fruit_b  
FROM basket_a a  
INNER JOIN basket_b b ON a.fruit = b.fruit;
```

Below the SQL Editor, there is an "Output pane" with four tabs: "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is active, showing the results of the query in a table format.

	id_tabelaa integer	fruit_tabelaa character varying(100)	id_tabelab integer	fruit_tabelab character varying(100)
1	1	Apple	2	Apple
2	2	Orange	1	Orange

Junções (JOINS)

INNER JOIN



INNER JOIN

Junções (JOINS)

LEFT OUTER JOIN

Retorna as linhas comuns às duas tabelas mais o conjunto de linhas não comuns da tabela da esquerda

Junções (JOINS)

LEFT OUTER JOIN

```
SELECT  a.id id_a,  
        a.fruit fruit_a,  
        b.id id_b,  
        b.fruit fruit_b  
FROM    basket_a a  
LEFT JOIN basket_b b ON a.fruit = b.fruit;
```


Junções (JOINS)

LEFT OUTER JOIN

SQL Editor

Graphical Query Builder

Previous queries

```
SELECT
    a.id id_a,
    a.fruit fruit_a,
    b.id id_b,
    b.fruit fruit_b
FROM
    basket_a a
LEFT JOIN basket_b b ON a.fruit = b.fruit;
```

Output pane

Data Output

Explain

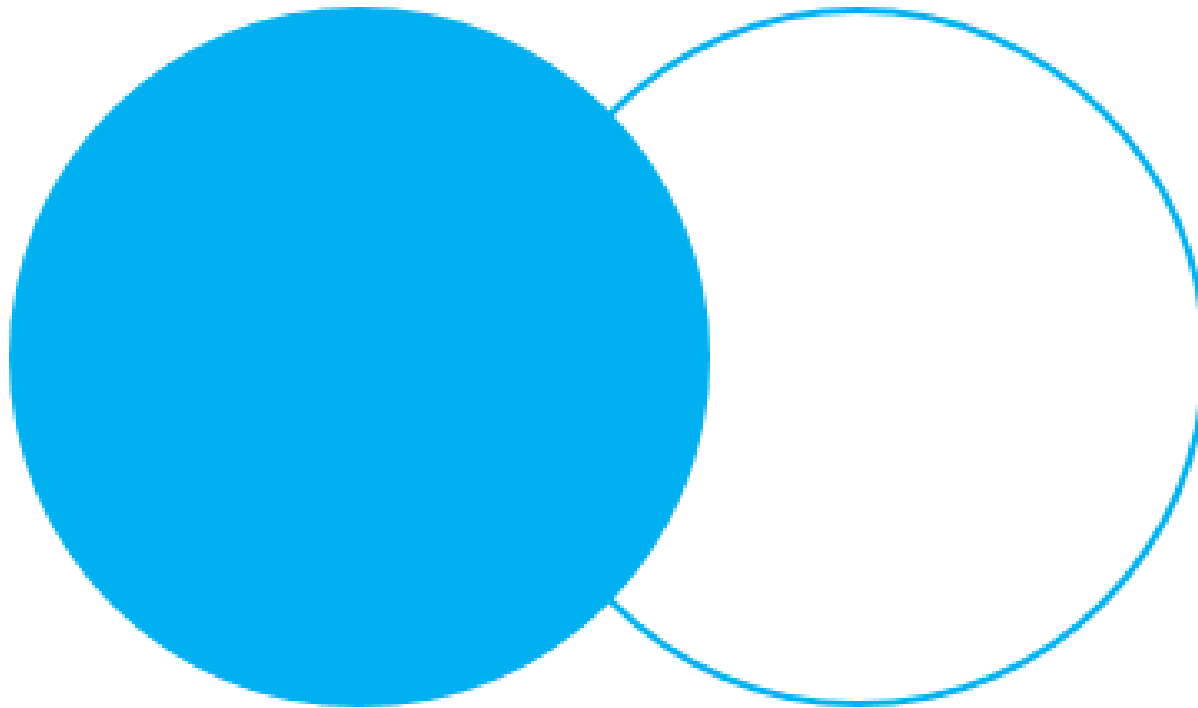
Messages

History

	id_a integer	fruit_a character varying(100)	id_b integer	fruit_b character varying(100)
1	1	Apple	2	Apple
2	2	Orange	1	Orange
3	3	Banana		
4	4	Cucumber		

Junções (JOINS)

LEFT OUTER JOIN



LEFT OUTER JOIN

Junções (JOINS)

LEFT OUTER JOIN - only

Retorna apenas o conjunto de instâncias não comuns na tabela à esquerda.

Deve ser usado com o WHERE

Junções (JOINS)

LEFT OUTER JOIN - only

```
SELECT  a.id id_a,  
        a.fruit fruit_a,  
        b.id id_b,  
        b.fruit fruit_b  
FROM    basket_a a  
LEFT JOIN basket_b b ON a.fruit = b.fruit  
WHERE b.id IS NULL;
```

Junções (JOINS)

LEFT OUTER JOIN - only

The screenshot shows a SQL Editor window with two tabs: "SQL Editor" and "Graphical Query Builder". The "SQL Editor" tab is active, displaying the following SQL query:

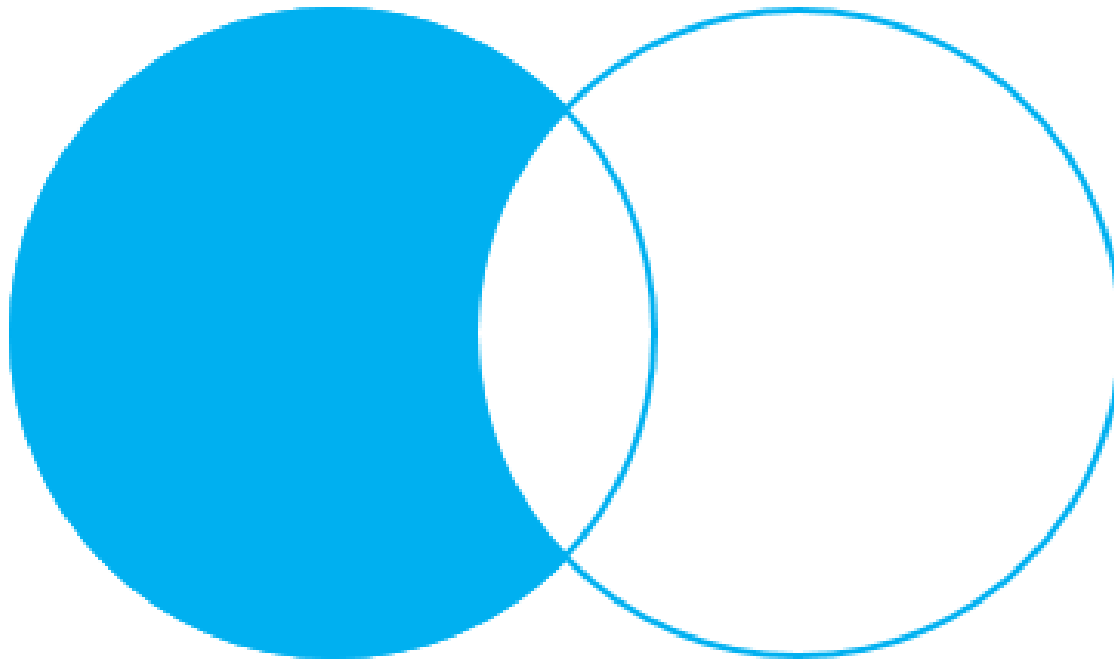
```
SELECT a.id id_a,  
       a.fruit fruit_a,  
       b.id id_b,  
       b.fruit fruit_b  
FROM   basket_a a  
LEFT JOIN basket_b b ON a.fruit = b.fruit  
WHERE b.id IS NULL;
```

Below the query editor is the "Output pane" with four tabs: "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is active, showing the results of the query in a table format.

	id_a integer	fruit_a character varying(100)	id_b integer	fruit_b character varying(100)
1	3	Banana		
2	4	Cucumber		

Junções (JOINS)

LEFT OUTER JOIN - only



LEFT OUTER JOIN – only
rows from the left table

Junções (JOINS)

RIGHT OUTER JOIN

Retorna as linhas comuns às duas tabelas mais as linhas não comuns que existem na tabela da direita.

Junções (JOINS)

RIGHT OUTER JOIN

```
SELECT  a.id id_a,  
        a.fruit fruit_a,  
        b.id id_b,  
        b.fruit fruit_b  
FROM    basket_a a  
RIGHT JOIN basket_b b ON a.fruit = b.fruit;
```


Junções (JOINS)

RIGHT OUTER JOIN

SQL Editor

Graphical Query Builder

Previous queries

```
SELECT  a.id id_a,
        a.fruit fruit_a,
        b.id id_b,
        b.fruit fruit_b
FROM    basket_a a
RIGHT JOIN basket_b b ON a.fruit = b.fruit;
```

Output pane

Data Output

Explain

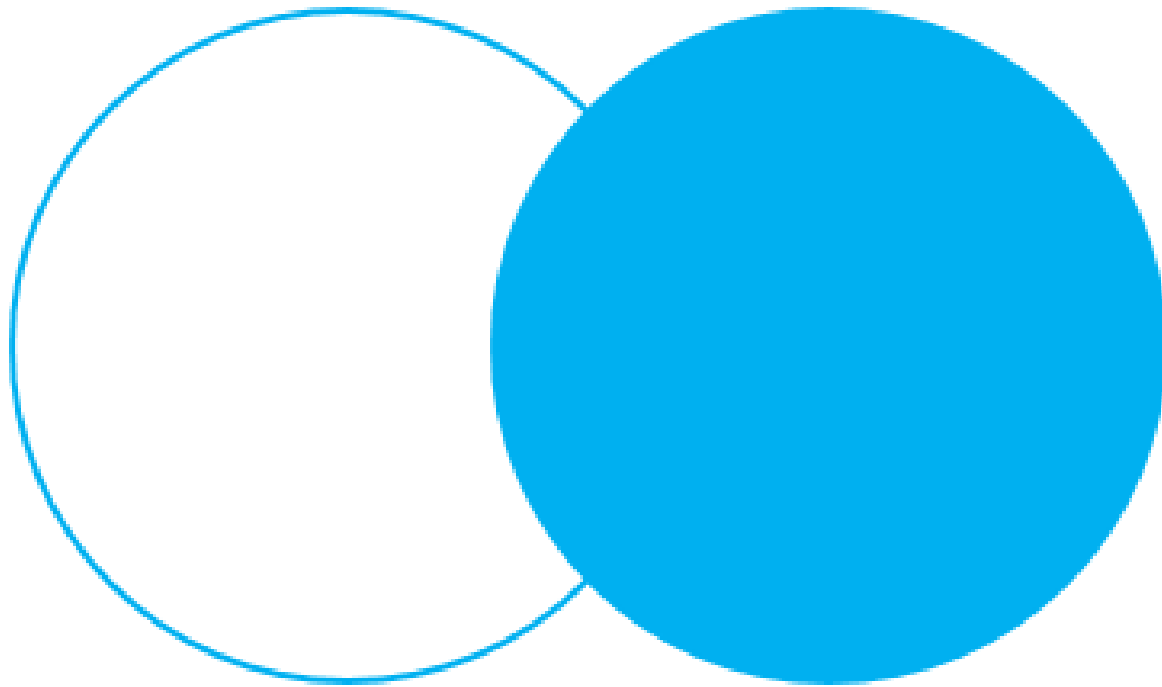
Messages

History

	id_a integer	fruit_a character varying(100)	id_b integer	fruit_b character varying(100)
1	2	Orange	1	Orange
2	1	Apple	2	Apple
3			3	Watermelon
4			4	Pear

Junções (JOINS)

RIGHT OUTER JOIN



RIGHT OUTER JOIN

Junções (JOINS)

RIGHT OUTER JOIN - only

Retorna apenas o conjunto de instâncias não comuns na tabela à direita.

Deve ser usado com o WHERE

Junções (JOINS)

RIGHT OUTER JOIN - only

```
SELECT  a.id id_a,  
        a.fruit fruit_a,  
        b.id id_b,  
        b.fruit fruit_b  
FROM    basket_a a  
RIGHT JOIN basket_b b ON a.fruit = b.fruit  
WHERE a.id IS NULL;
```

Junções (JOINS)

RIGHT OUTER JOIN - only

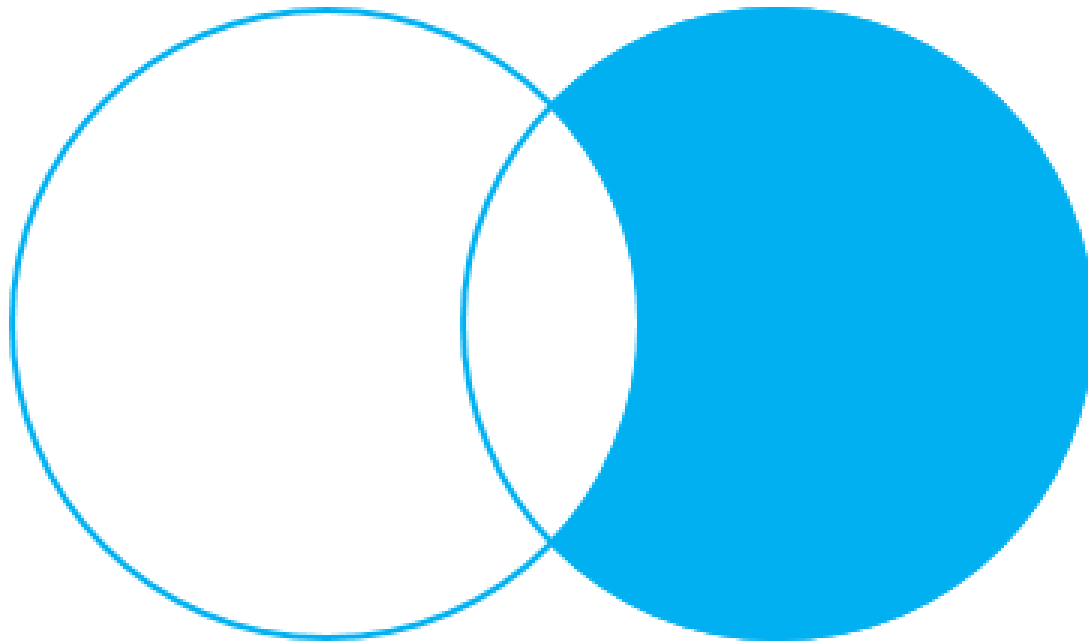
The screenshot shows a database management system interface. At the top, there are two tabs: 'SQL Editor' and 'Graphical Query Builder'. Below these is a 'Previous queries' section. The main area displays a SQL query in the 'SQL Editor' tab. The query is a RIGHT OUTER JOIN between two tables, 'basket_a' and 'basket_b', on the condition 'a.fruit = b.fruit'. The query also includes a WHERE clause 'a.id IS NULL;'. Below the query editor is an 'Output pane' with four tabs: 'Data Output', 'Explain', 'Messages', and 'History'. The 'Data Output' tab is selected, showing a table with two columns: 'id_a' (integer) and 'fruit_a' (character varying(100)). The table has two rows of data: row 1 with 'id_a' 3 and 'fruit_a' 'Watermelon', and row 2 with 'id_a' 4 and 'fruit_a' 'Pear'.

```
SELECT a.id id_a,  
       a.fruit fruit_a,  
       b.id id_b,  
       b.fruit fruit_b  
FROM   basket_a a  
RIGHT JOIN basket_b b ON a.fruit = b.fruit  
WHERE  a.id IS NULL;
```

	id_a integer	fruit_a character varying(100)	id_b integer	fruit_b character varying(100)
1			3	Watermelon
2			4	Pear

Junções (JOINS)

RIGHT OUTER JOIN - only



RIGHT OUTER JOIN – only
rows from the right table

Junções (JOINS)

FULL OUTER JOIN

Retorna todas as linhas comuns às tabelas (as associadas) mais as linhas não comuns que existem nas duas tabelas

Junções (JOINS)

FULL OUTER JOIN

```
SELECT  a.id id_a,  
        a.fruit fruit_a,  
        b.id id_b,  
        b.fruit fruit_b  
FROM    basket_a a  
FULL OUTER JOIN basket_b b ON a.fruit = b.fruit;
```


Junções (JOINS)

FULL OUTER JOIN

SQL Editor Graphical Query Builder

Previous queries

```
SELECT  a.id id_a,  
        a.fruit fruit_a,  
        b.id id_b,  
        b.fruit fruit_b  
FROM    basket_a a  
FULL OUTER JOIN basket_b b ON a.fruit = b.fruit;
```

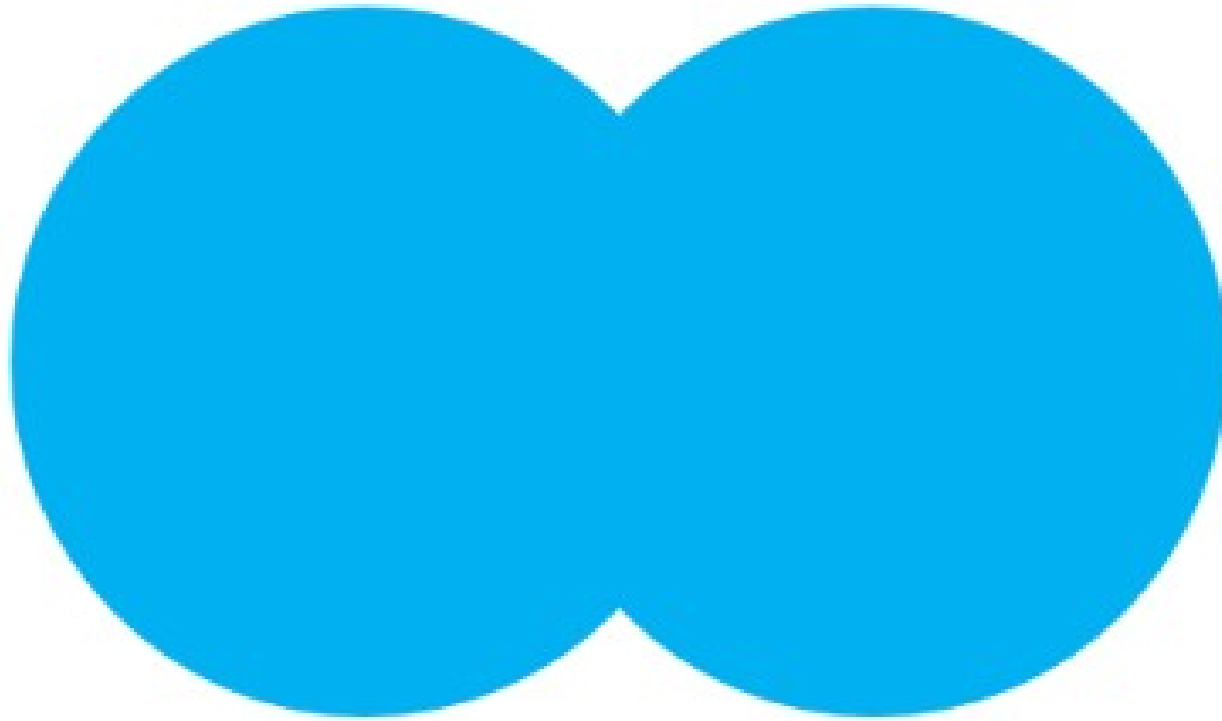
Output pane

Data Output Explain Messages History

	id_a integer	fruit_a character varying(100)	id_b integer	fruit_b character varying(100)
1	1	Apple	2	Apple
2	2	Orange	1	Orange
3	3	Banana		
4	4	Cucumber		
5			3	Watermelon
6			4	Pear

Junções (JOINS)

FULL OUTER JOIN



FULL OUTER JOIN

Junções (JOINS)

FULL OUTER JOIN

Retorna todas as linhas NÃO comuns entre as tabelas (não associadas).

Junções (JOINS)

FULL OUTER JOIN - only

```
SELECT  a.id id_a,  
        a.fruit fruit_a,  
        b.id id_b,  
        b.fruit fruit_b  
FROM    basket_a a  
FULL JOIN basket_b b ON a.fruit = b.fruit  
WHERE a.id IS NULL OR b.id IS NULL;
```

Junções (JOINS)

FULL OUTER JOIN - only

SQL Editor Graphical Query Builder

Previous queries

```
SELECT  a.id id_a,
        a.fruit fruit_a,
        b.id id_b,
        b.fruit fruit_b
FROM    basket_a a
FULL JOIN basket_b b ON a.fruit = b.fruit
WHERE a.id IS NULL OR b.id IS NULL;
```

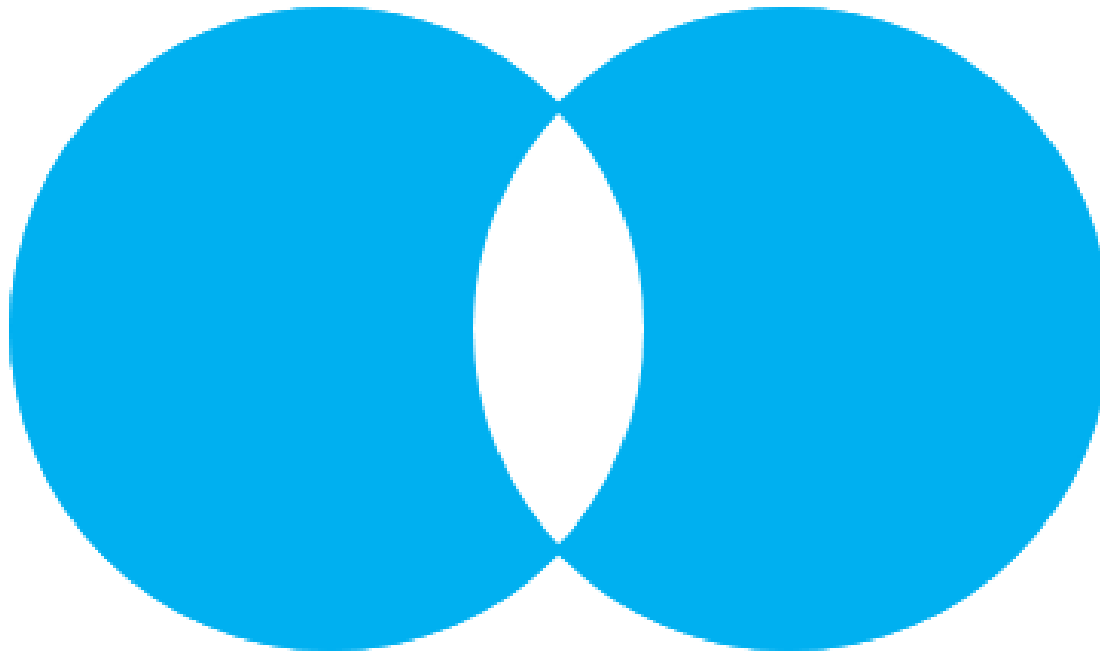
Output pane

Data Output Explain Messages History

	id_a integer	fruit_a character varying(100)	id_b integer	fruit_b character varying(100)
1	3	Banana		
2	4	Cucumber		
3			3	Watermelon
4			4	Pear

Junções (JOINS)

FULL OUTER JOIN - only



FULL OUTER JOIN – only
rows unique to both tables

Atividade

Crie uma base de dados chamada exercicioJOIN

Rode o script “atividade.sql” (GITHUB)

Responda as perguntas da Atividade Joins.

Envie pelo classroom até terça-feira.

Dúvidas?

Alanamm.prof@gmail.com