

INSERT INTO (inserção)

- EXEMPLO: Inserir uma nova tupla (linha) na tabela Peca.

```
INSERT INTO Peca (Cod_Peca, Nome_Peca, Preco,  
Qte)
```

```
VALUES (380, 'Peca W', 77.00, 23) //caracteres entre  
aspas duplas
```

ATENÇÃO: NULL e valore DEFAULT

DELETE FROM (exclusão)

- EXEMPLO: Excluir a peça com código 200 (**toda a linha**).

```
DELETE FROM Peca  
WHERE Cod_Peca = 200
```

ATENÇÃO: Não se esqueça da cláusula WHERE

UPDATE/ SET (alteração)

- EXEMPLO: Alterar o Preço da peça de código 200 para 90.00
- UPDATE Peca
SET Preço = 90.00
WHERE Cod_Peca = 200

ATENÇÃO: Não se esqueça da cláusula WHERE

SELECT (selecionar)

- EXEMPLO: Selecionar o código e o nome das peças com preço menor do que 70.00:

```
SELECT Cod_Peca, Nome_Peca  
FROM Peca  
WHERE Preço < 70.00
```

SELECT (selecionar)

CLÁUSULA WHERE

- A cláusula **WHERE** especifica as condições que devem ser satisfeitas.
- Usa conectores lógicos:
 - AND (e)
 - OR (ou)
 - NOT (não)

SELECT (selecionar)

CLÁUSULA WHERE

- Usa operadores de comparação:
 - > (maior)
 - < (menor)
 - = (igual)
 - <= (menor ou igual)
 - >= (maior ou igual)
 - BETWEEN (entre um intervalo, **incluindo os extremos**)
 - facilita a especificação de condições numéricas que envolvam um intervalo, ao invés de usar os operadores =< e >=.

SELECT (selecionar) – ORDER BY

CLÁUSULA ORDER BY (ORDENAÇÃO E APRESENTAÇÃO DE TUPLAS)

- Aplicado apenas à operação **SELECT**, depois da cláusula **WHERE**
- A cláusula **ORDER BY** faz com que as tuplas do resultado de uma consulta apareçam em uma determinada ordem.

SELECT (selecionar) – ORDER BY

- Para especificar a forma de ordenação, devemos indicar:
 - **Desc**: ordem decrescente (decrecente)
 - **Asc**: ordem ascendente (crescente)**default**
- **Sintaxe:**

```
ORDER BY nome_atributo ASC  
ORDER BY nome_atributo DESC
```


FUNÇÕES DE AGREGAÇÃO

- As funções de agregação servem para recuperar dados agregados, ou seja, dados que foram trabalhados sobre os dados armazenados.
- As principais são:
 - **SUM** (soma dos valores da coluna – tipo de dado numérico),
 - **MAX** (valor máximo),
 - **MIN** (valor mínimo),
 - **AVG** (average - média – deve ser numérico),
 - **COUNT** (contador de tuplas - as linhas - da relação).
- **ATENÇÃO: SUM \neq COUNT**

SQL – DML





*(Data Manipulation Definition -
linguagem de manipulação de
dados)*

Parte 2











Valores NULL

- Suponha que temos a tabela peca criada na última aula preenchida até o momento da seguinte forma

	cod_peca [PK] integer 	nome_peca character varying (30) 	preco numeric (6,2) 	qte integer 
1	1	Peca A	15.00	10
2	2	Peca B	8.00	20
3	3	Peca C	8.00	30
4	4	Peca B	8.00	10

Valores NULL

- Suponha que temos a tabela peca criada na última aula preenchida até o momento da seguinte forma

Columns								+
		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
		cod_pec	integer v			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
		nome_pec	character varying v	30		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
		preco	numeric v	6	2	<input type="checkbox"/>	<input type="checkbox"/>	
		qte	integer v			<input type="checkbox"/>	<input type="checkbox"/>	0

Inserindo Valores NULOS

- Quando realizamos um INSERT e não passamos o campo, o banco de dados vai automaticamente inserir NULL no valor da célula.

Inserindo Valores NULOS

Query

Query History

```
1 INSERT INTO peca (cod_pecas, nome_pecas, qte)
2 VALUES (5, 'Peca E', 15);
3 SELECT * FROM peca;
```

Data Output

Messages

Notifications

≡+

📄

▼

📋

🗑️

🗄️

⬇️

📈

	<div><div>cod_pecas</div><div>[PK] integer</div></div>	<div><div>nome_pecas</div><div>character varying (30)</div></div>	<div><div>preco</div><div>numeric (6,2)</div></div>	<div><div>qte</div><div>integer</div></div>
1	1	Peca A	15.00	10
2	2	Peca B	8.00	20
3	3	Peca C	8.00	30
4	4	Peca B	8.00	10
5	5	Peca E	[null]	15

CUIDADO com o DEFAULT

- Lembre-se que colunas que tem o valor DEFAULT definido, não serão preenchidas com NULL, mas sim com o valor DEFAULT

CUIDADO com o DEFAULT

Query

Query History

1

INSERT INTO peca (cod_pecas, nome_pecas, preco)

2

VALUES (6, 'Peca F', 17.00);

3

SELECT * FROM peca;

Data Output

Messages

Notifications

≡+

📄

✓

📋

🗑

🗄

⬇

📈

	<div><div>cod_pecas</div><div>[PK] integer</div></div>	<div><div>nome_pecas</div><div>character varying (30)</div></div>	<div><div>preco</div><div>numeric (6,2)</div></div>	<div><div>qte</div><div>integer</div></div>
1	1	Peca A	15.00	10
2	2	Peca B	8.00	20
3	3	Peca C	8.00	30
4	4	Peca B	8.00	10
5	5	Peca E	[null]	15
6	6	Peca F	17.00	0

Inserindo Valor NULL

- Existe uma outra forma de definir um valor como NULL. Essa forma é deixando explícito no comando que a coluna deve receber valor NULL

Inserindo Valor NULL

Query

Query History

1

2

3

INSERT INTO

peca

(cod_peca, nome_peca, preco, qte)

VALUES (7, 'Peca G', 17.00, NULL);

SELECT * FROM peca;

Data Output

Messages

Notifications

≡+

📄

▼

📋

🗑️

🗄️

⬇️

📈

	<div><div>cod_peca</div><div>[PK] integer</div></div>	<div><div>nome_peca</div><div>character varying (30)</div></div>	<div><div>preco</div><div>numeric (6,2)</div></div>	<div><div>qte</div><div>integer</div></div>
1	1	Peca A	15.00	10
2	2	Peca B	8.00	20
3	3	Peca C	8.00	30
4	4	Peca B	8.00	10
5	5	Peca E	[null]	15
6	6	Peca F	17.00	0
7	7	Peca G	17.00	[null]

Inserindo valor NULL

- Repare, que mesmo o campo QTE possuindo um valor DEFAULT, foi definido de forma explícita no INSERT que essa coluna deveria possuir um valor NULL

ATENÇÃO

- Se você tentar inserir o valor NULL em uma coluna definida como NOT NULL, uma exceção (ERRO) será lançada pelo banco de dados.

Query	Query History	Scratch Pad ×
1	<code>INSERT INTO peca (cod_pecas, nome_pecas, preco, qte)</code>	
2	<code>VALUES (7, NULL, 17.00, 12);</code>	

Data Output	Messages	Notifications
<p>ERROR: null value in column "nome_pecas" of relation "pecas" violates not-null constraint DETAIL: Failing row contains (7, null, 17.00, 12). SQL state: 23502</p>		

Selecionando Valores Nulos

- Caso você queira retornar somente as linhas que possuam valores NULOS em uma determinada célula, a forma correta é utilizar o IS NULL e não = NULL

FORMA CORRETA

The screenshot shows a database query tool interface. At the top, there are two tabs: "Query" and "Query History". The "Query" tab is active, displaying a SQL query with line numbers 1 and 2. Below the query, there are three tabs: "Data Output", "Messages", and "Notifications". The "Data Output" tab is active, showing a table of results. Above the table is a toolbar with icons for various actions like adding, saving, and deleting. The table has five columns: an index column, "cod_peca", "nome_peca", "preco", and "qte". The first row of data shows the value 1 in the index column, 5 in the "cod_peca" column, "Peca E" in the "nome_peca" column, [null] in the "preco" column, and 15 in the "qte" column.

```
1 SELECT * FROM peca
2 WHERE preco IS NULL;
```

	cod_peca [PK] integer	nome_peca character varying (30)	preco numeric (6,2)	qte integer
1	5	Peca E	[null]	15

FORMA INCORRETA

The screenshot shows a database query editor interface. The 'Query' tab is active, displaying a SQL query with two lines: '1 SELECT * FROM peca' and '2 WHERE preco = NULL;'. The 'Data Output' tab is also visible, showing a table with four columns: 'cod_eca' (integer, primary key), 'nome_eca' (character varying (30)), 'preco' (numeric (6,2)), and 'qte' (integer). Each column name is followed by a pencil icon, indicating it can be edited. The interface includes a toolbar with icons for menu, save, undo, redo, delete, refresh, and zoom.

Query Query History

```
1 SELECT * FROM peca
2 WHERE preco = NULL;
```

Data Output Messages Notifications

	cod_eca [PK] integer	nome_eca character varying (30)	preco numeric (6,2)	qte integer		

Selecionando Valores NÃO Nulos

- Caso você queira selecionar somente as linhas que NÃO possuam valor nulo em uma determinada coluna, é só utilizar o comando `IS NOT NULL`

Selecionando Valores NÃO Nulos

Query

Query History

1

SELECT * FROM peca

2

WHERE preco IS NOT NULL;

Data Output

Messages

Notifications

	<div><div>cod_pec</div><div>[PK] integer</div></div>	<div><div>nome_pec</div><div>character varying (30)</div></div>	<div><div>preco</div><div>numeric (6,2)</div></div>	<div><div>qte</div><div>integer</div></div>
1	1	Peca A	15.00	10
2	2	Peca B	8.00	20
3	3	Peca C	8.00	30
4	4	Peca B	8.00	10
5	6	Peca F	17.00	0
6	7	Peca G	17.00	[null]

Ordenando colunas com NULL

- Por default, caso você ordene um SELECT por uma coluna que possui células com valor NULL, essas células serão as últimas a serem retornadas

Ordenando colunas com NULL

Query

Query History

1

SELECT * FROM peca

2

ORDER BY preco;

Data Output

Messages

Notifications

	<div><div>cod_peca</div><div>[PK] integer</div></div>	<div><div>nome_peca</div><div>character varying (30)</div></div>	<div><div>preco</div><div>numeric (6,2)</div></div>	<div><div>qte</div><div>integer</div></div>
1	4	Peca B	8.00	10
2	2	Peca B	8.00	20
3	3	Peca C	8.00	30
4	1	Peca A	15.00	10
5	6	Peca F	17.00	0
6	7	Peca G	17.00	[null]
7	5	Peca E	[null]	15

Ordenando colunas com NULL

- Caso você deseje que as células com valores NULL sejam as primeiras a serem retornadas no SELECT, utiliza o ORDER BY ... NULLS FIRST;

Ordenando colunas com NULL

Query

Query History

1

SELECT * FROM peca

2

ORDER BY preco NULLS FIRST;

Data Output

Messages

Notifications

	<div><div>cod_pec</div><div>[PK] integer</div></div>	<div><div>nome_pec</div><div>character varying (30)</div></div>	<div><div>preco</div><div>numeric (6,2)</div></div>	<div><div>qte</div><div>integer</div></div>
1	5	Peca E	[null]	15
2	2	Peca B	8.00	20
3	3	Peca C	8.00	30
4	4	Peca B	8.00	10
5	1	Peca A	15.00	10
6	6	Peca F	17.00	0
7	7	Peca G	17.00	[null]

SELECT (selecionar)

DISTINCT

- Tuplas duplicadas podem aparecer nas relações.
- No caso de desejarmos a eliminação de duplicidade, devemos inserir a palavra **DISTINCT** na cláusula **SELECT**.

Observações:

- Funções agregadas normalmente consideram as tuplas duplicadas.
- Não é permitido o uso do **DISTINCT** com o **COUNT(*)**.
- É válido usar o **DISTINCT** com **MAX** ou **MIN**, mesmo não alterando o resultado.

Tabela Neste Momento

Query

Query History

1

SELECT * FROM peca;

Data Output

Messages

Notifications

	<div>cod_peca</div> <div>[PK] integer</div>	<div>nome_peca</div> <div>character varying (30)</div>	<div>preco</div> <div>numeric (6,2)</div>	<div>qte</div> <div>integer</div>
1	1	Peca A	15.00	10
2	2	Peca B	8.00	20
3	4	Peca B	8.00	10
4	3	Peca A	8.00	30
5	6	Peca C	17.00	0
6	7	Peca C	17.00	[null]
7	5	Peca A	[null]	15

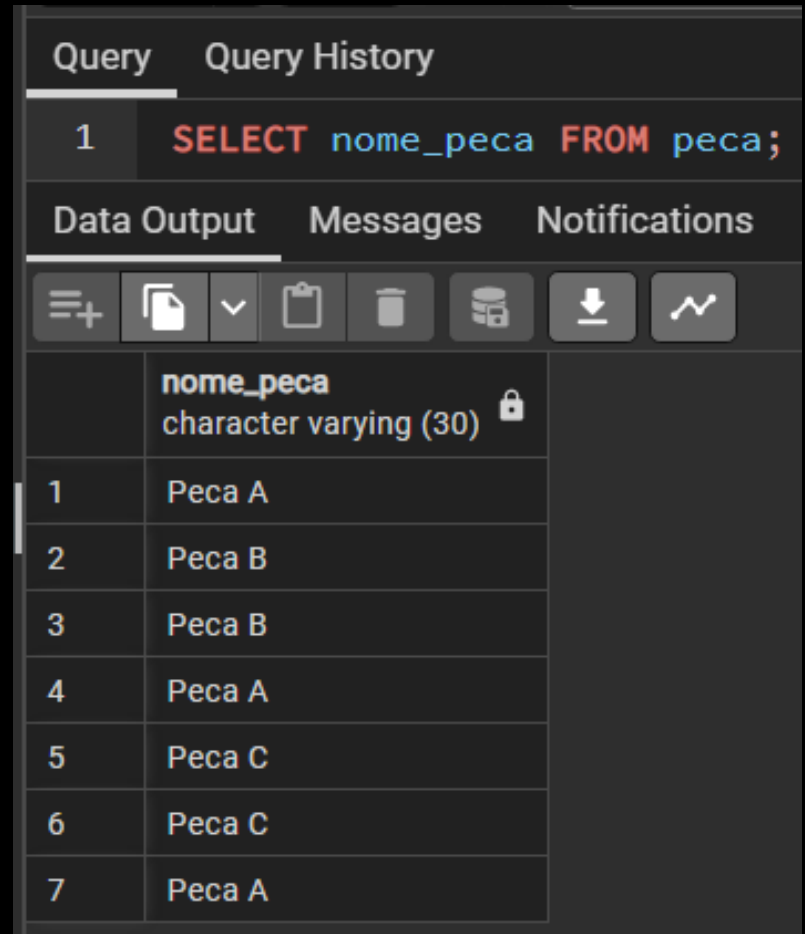
SELECT (selecionar)

DISTINCT


Exemplo1: Obter o nome de todas peças (sem distinct).

```
SELECT nome_peca
```

```
FROM peca
```



The screenshot shows a database query interface with a dark theme. At the top, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query: `1 SELECT nome_peca FROM peca;`. Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with the results of the query. The table has two columns: 'nome_peca' (character varying (30)) and a lock icon. The results are as follows:

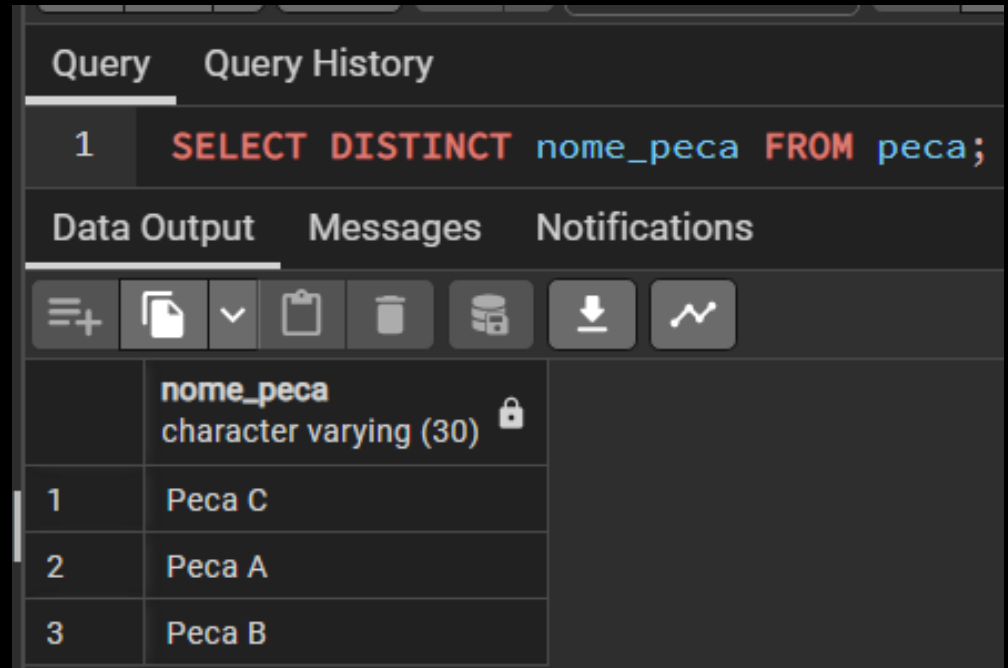
	nome_peca character varying (30) 
1	Peca A
2	Peca B
3	Peca B
4	Peca A
5	Peca C
6	Peca C
7	Peca A

SELECT (selecionar)

DISTINCT

- **Exemplo2:** Obter o nome de todas as peças (usando a cláusula distinct).

```
SELECT  
DISTINCT nome_peca  
FROM peca;
```



The screenshot shows a database query tool interface. The 'Query' tab is active, displaying the SQL query: `SELECT DISTINCT nome_peca FROM peca;`. Below the query, the 'Data Output' tab is selected, showing the results of the query. The results are displayed in a table with two columns: an index and the piece name. The table contains three rows of data: 'Peca C', 'Peca A', and 'Peca B'.

	nome_peca character varying (30)
1	Peca C
2	Peca A
3	Peca B

SELECT (selecionar)

CLÁUSULA GROUP BY = agrupar por

- São funções que tomam uma coleção (conjunto ou subconjunto) de valores como entrada, retornando um valor simples.
- Só podem ser usadas em comandos SELECT.
- Normalmente utilizada em conjunto com as funções de agregação (COUNT, SUM, AVG, MIN, MAX).
- Agrupa as tuplas selecionadas com base em um de seus atributos (quando este atributo é igual em duas ou mais tuplas elas são agrupadas).
- Desta forma, a função de agregação será aplicada a cada grupo, e não a todas as tuplas.

Tabela Neste Momento

```
1 SELECT * FROM peca ORDER BY cod_pecas;
```

Data Output Messages Notifications



	cod_pecas [PK] integer	nome_pecas character varying (30)	preco numeric (6,2)	qte integer	veiculo character varying (8)
1	1	Peca A	15.00	10	CARRO
2	2	Peca B	8.00	20	MOTO
3	3	Peca C	8.00	30	CAMINHAO
4	4	Peca D	8.00	10	CARRO
5	5	Peca E	[null]	15	CAMINHAO
6	6	Peca F	17.00	0	MOTO
7	7	Peca G	17.00	[null]	CARRO

SELECT (selecionar)

CLÁUSULA GROUP BY = agrupar por

- **Exemplo1:** Obter o número de peças cadastradas.

```
SELECT COUNT (*)  
FROM pecas
```

- **RESULTADO**
(contagem de tuplas):

Query

Query History

1

SELECT COUNT(*) FROM peca;

Data Output

Messages

Notifications

	count bigint	
1		7

SELECT (selecionar)

CLÁUSULA GROUP BY = agrupar por

- **Exemplo 2:** Obter o nº de peças por veículo //contar por grupos

Query

Query History

1

SELECT veiculo, COUNT (1) FROM peca GROUP BY veiculo;

Data Output

Messages

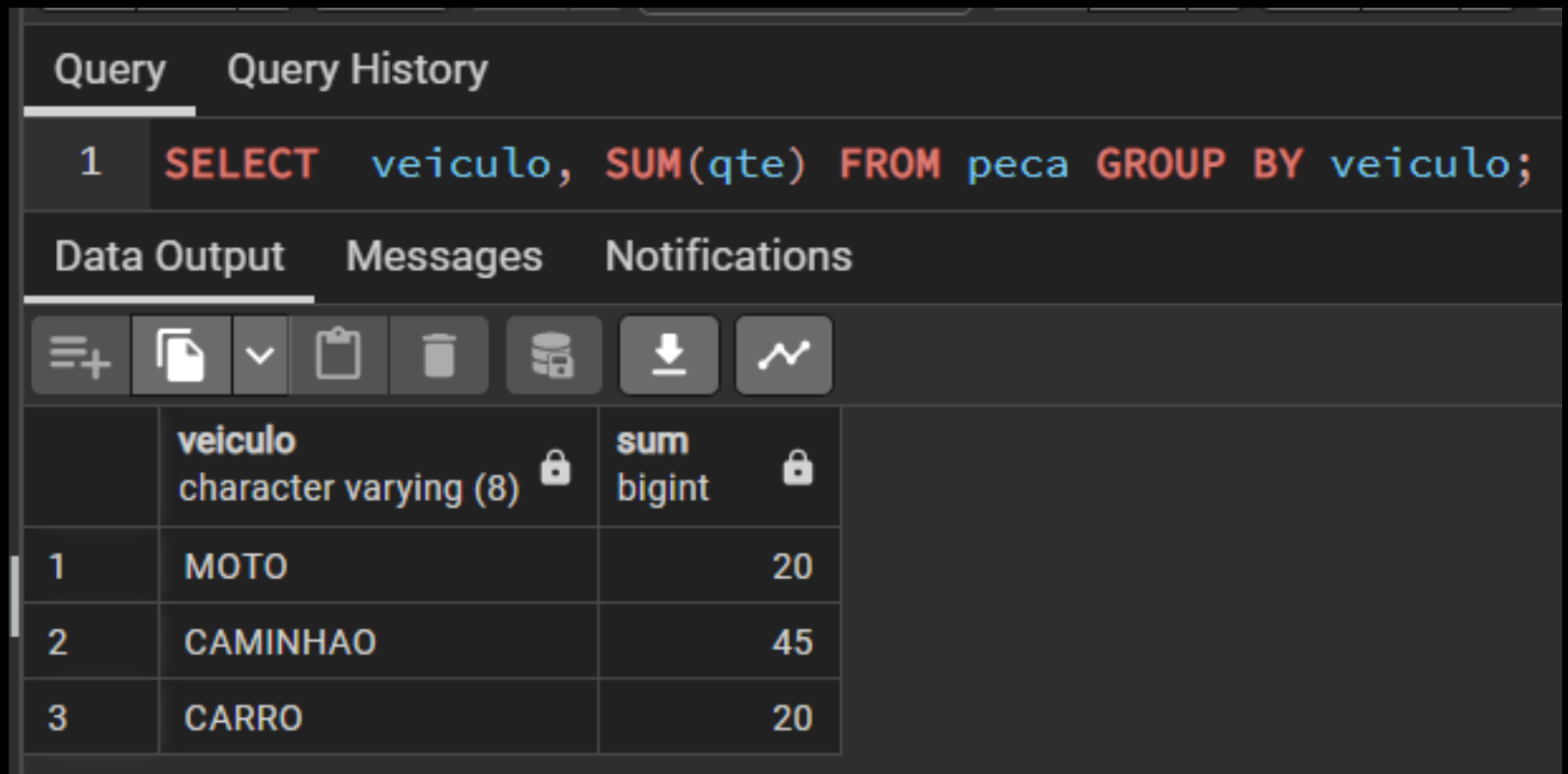
Notifications

	veiculo character varying (8)	count bigint
1	MOTO	2
2	CAMINHAO	2
3	CARRO	3

SELECT (selecionar)

CLÁUSULA GROUP BY = agrupar por

- Obter a soma da quantidade de peças por tipo de veículo.





The screenshot shows a database query editor interface. At the top, there are tabs for 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL query: `1 SELECT veiculo, SUM(qte) FROM peca GROUP BY veiculo;`. Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with the results of the query. The table has two columns: 'veiculo' (character varying (8)) and 'sum' (bigint). The results are as follows:

	veiculo character varying (8)	sum bigint
1	MOTO	20
2	CAMINHAO	45
3	CARRO	20

SELECT (selecionar)

CLÁUSULA HAVING

- É utilizada para filtrar as linhas do grupo criado por GROUP BY.
- **Exemplo 2 (anterior) alterado:** obter a soma da quantidade de peças que sejam maiores que 20.

Query		Query History	
1	SELECT	veiculo,	SUM(qte)
2	FROM	peca	
3	GROUP BY	veiculo	HAVING SUM(qte) > 20;
Data Output		Messages	Notifications
	veiculo		sum
	character varying (8) 		bigint 
1	CAMINHAO		45