



Estácio

CAMPUS: Polo Jacareí - SP – Centro

CURSO: Desenvolvimento FullStack

DISCIPLINA: Vamos manter as informações!

TURMA: 2025.1

SEMESTRE LETIVO: Primeiro Semestre (2025)

ALUNO: Maurício Pereira Campos

MATRÍCULA: 202403843447

Título da prática:

Modelagem e implementação de um banco de dados simples, utilizando como base o SQL Server.

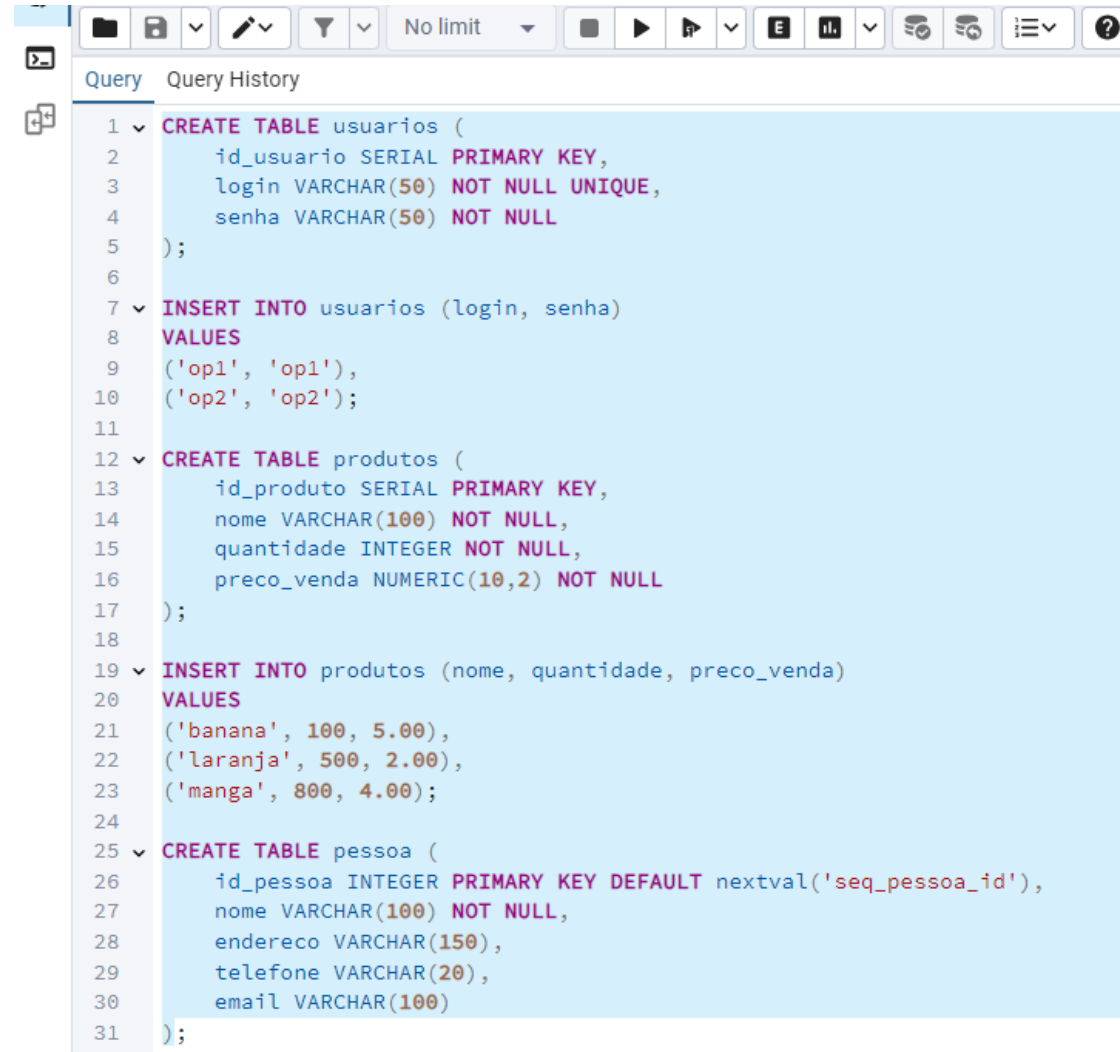
Objetivos da prática:

1. Identificar os requisitos de um sistema e transformá-los no modelo adequado.
2. Utilizar ferramentas de modelagem para bases de dados relacionais.
3. Explorar a sintaxe SQL na criação das estruturas do banco (DDL).
4. Explorar a sintaxe SQL na consulta e manipulação de dados (DML)

Segundo Procedimento:

Alimentando a Base

Códigos usados neste roteiro

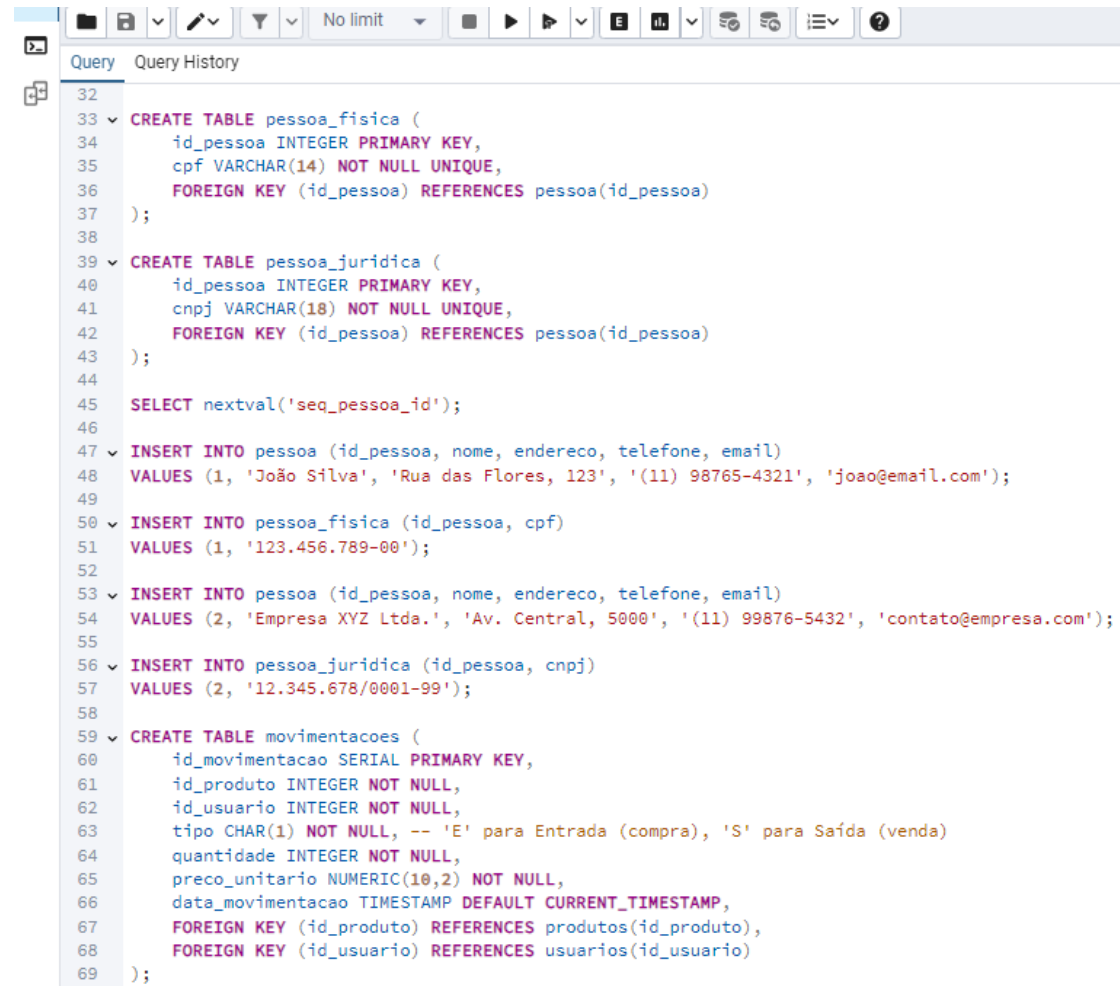


```
1 CREATE TABLE usuarios (  
2     id_usuario SERIAL PRIMARY KEY,  
3     login VARCHAR(50) NOT NULL UNIQUE,  
4     senha VARCHAR(50) NOT NULL  
5 );  
6  
7 INSERT INTO usuarios (login, senha)  
8 VALUES  
9 ('op1', 'op1'),  
10 ('op2', 'op2');  
11  
12 CREATE TABLE produtos (  
13     id_produto SERIAL PRIMARY KEY,  
14     nome VARCHAR(100) NOT NULL,  
15     quantidade INTEGER NOT NULL,  
16     preco_venda NUMERIC(10,2) NOT NULL  
17 );  
18  
19 INSERT INTO produtos (nome, quantidade, preco_venda)  
20 VALUES  
21 ('banana', 100, 5.00),  
22 ('laranja', 500, 2.00),  
23 ('manga', 800, 4.00);  
24  
25 CREATE TABLE pessoa (  
26     id_pessoa INTEGER PRIMARY KEY DEFAULT nextval('seq_pessoa_id'),  
27     nome VARCHAR(100) NOT NULL,  
28     endereco VARCHAR(150),  
29     telefone VARCHAR(20),  
30     email VARCHAR(100)  
31 );
```

Segundo Procedimento:

Alimentando a Base

Códigos usados neste roteiro

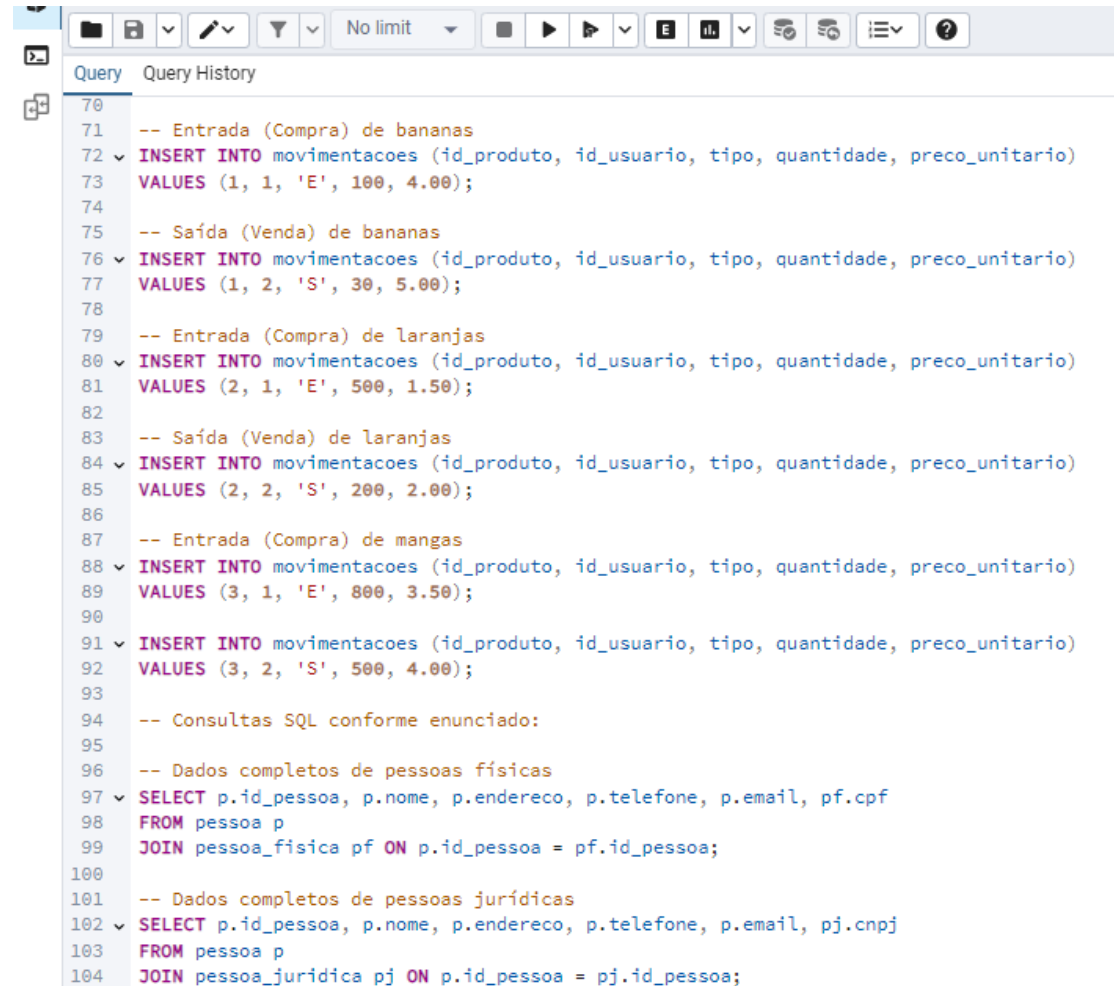


```
32
33 CREATE TABLE pessoa_fisica (
34     id_pessoa INTEGER PRIMARY KEY,
35     cpf VARCHAR(14) NOT NULL UNIQUE,
36     FOREIGN KEY (id_pessoa) REFERENCES pessoa(id_pessoa)
37 );
38
39 CREATE TABLE pessoa_juridica (
40     id_pessoa INTEGER PRIMARY KEY,
41     cnpj VARCHAR(18) NOT NULL UNIQUE,
42     FOREIGN KEY (id_pessoa) REFERENCES pessoa(id_pessoa)
43 );
44
45 SELECT nextval('seq_pessoa_id');
46
47 INSERT INTO pessoa (id_pessoa, nome, endereco, telefone, email)
48 VALUES (1, 'João Silva', 'Rua das Flores, 123', '(11) 98765-4321', 'joao@email.com');
49
50 INSERT INTO pessoa_fisica (id_pessoa, cpf)
51 VALUES (1, '123.456.789-00');
52
53 INSERT INTO pessoa (id_pessoa, nome, endereco, telefone, email)
54 VALUES (2, 'Empresa XYZ Ltda.', 'Av. Central, 5000', '(11) 99876-5432', 'contato@empresa.com');
55
56 INSERT INTO pessoa_juridica (id_pessoa, cnpj)
57 VALUES (2, '12.345.678/0001-99');
58
59 CREATE TABLE movimentacoes (
60     id_movimentacao SERIAL PRIMARY KEY,
61     id_produto INTEGER NOT NULL,
62     id_usuario INTEGER NOT NULL,
63     tipo CHAR(1) NOT NULL, -- 'E' para Entrada (compra), 'S' para Saída (venda)
64     quantidade INTEGER NOT NULL,
65     preco_unitario NUMERIC(10,2) NOT NULL,
66     data_movimentacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
67     FOREIGN KEY (id_produto) REFERENCES produtos(id_produto),
68     FOREIGN KEY (id_usuario) REFERENCES usuarios(id_usuario)
69 );
```

Segundo Procedimento:

Alimentando a Base

Códigos usados neste roteiro

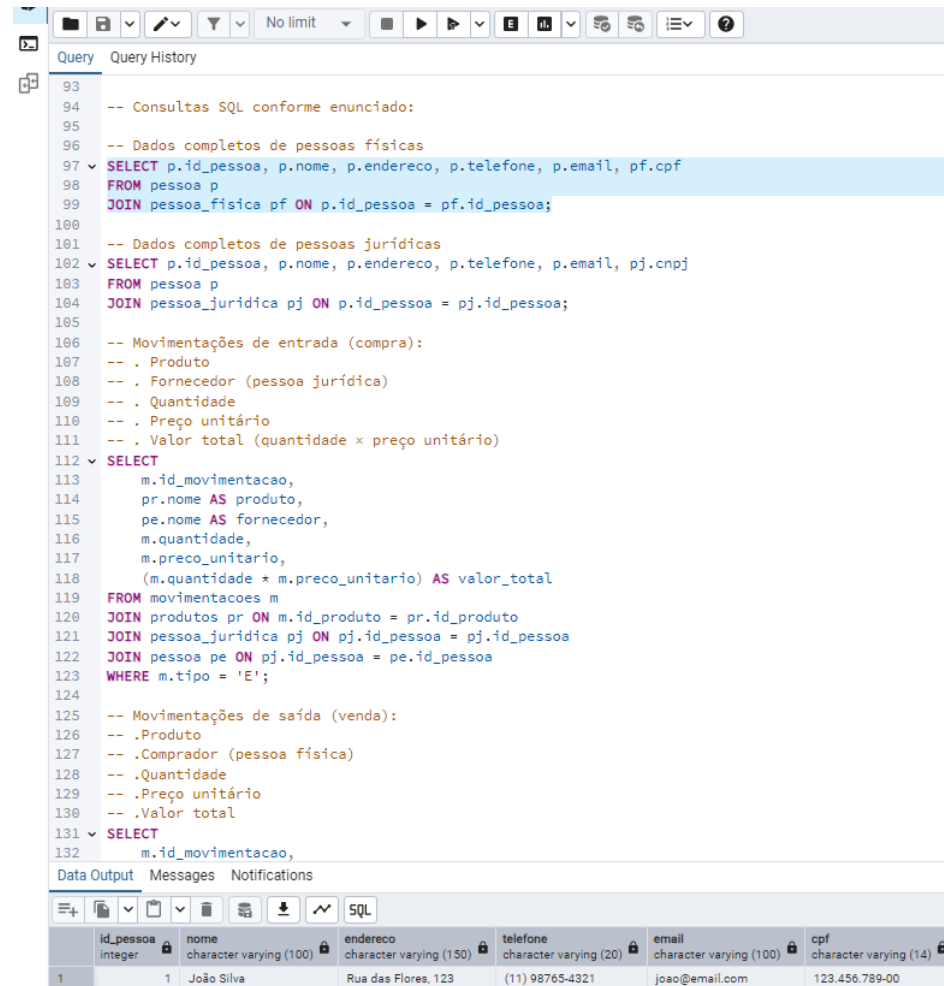


```
70
71 -- Entrada (Compra) de bananas
72 v INSERT INTO movimentacoes (id_produto, id_usuario, tipo, quantidade, preco_unitario)
73 VALUES (1, 1, 'E', 100, 4.00);
74
75 -- Saída (Venda) de bananas
76 v INSERT INTO movimentacoes (id_produto, id_usuario, tipo, quantidade, preco_unitario)
77 VALUES (1, 2, 'S', 30, 5.00);
78
79 -- Entrada (Compra) de laranjas
80 v INSERT INTO movimentacoes (id_produto, id_usuario, tipo, quantidade, preco_unitario)
81 VALUES (2, 1, 'E', 500, 1.50);
82
83 -- Saída (Venda) de laranjas
84 v INSERT INTO movimentacoes (id_produto, id_usuario, tipo, quantidade, preco_unitario)
85 VALUES (2, 2, 'S', 200, 2.00);
86
87 -- Entrada (Compra) de mangas
88 v INSERT INTO movimentacoes (id_produto, id_usuario, tipo, quantidade, preco_unitario)
89 VALUES (3, 1, 'E', 800, 3.50);
90
91 v INSERT INTO movimentacoes (id_produto, id_usuario, tipo, quantidade, preco_unitario)
92 VALUES (3, 2, 'S', 500, 4.00);
93
94 -- Consultas SQL conforme enunciado:
95
96 -- Dados completos de pessoas físicas
97 v SELECT p.id_pessoa, p.nome, p.endereco, p.telefone, p.email, pf.cpf
98 FROM pessoa p
99 JOIN pessoa_fisica pf ON p.id_pessoa = pf.id_pessoa;
100
101 -- Dados completos de pessoas jurídicas
102 v SELECT p.id_pessoa, p.nome, p.endereco, p.telefone, p.email, pj.cnpj
103 FROM pessoa p
104 JOIN pessoa_juridica pj ON p.id_pessoa = pj.id_pessoa;
```

Segundo Procedimento:

Alimentando a Base

Códigos usados neste roteiro



The screenshot displays a SQL IDE interface. The top toolbar includes icons for file operations, query execution, and settings. Below the toolbar, the 'Query' tab is active, showing a SQL script with line numbers 93 to 132. The script contains comments and SQL queries for inserting data into a database. The first query selects data from 'pessoa' and 'pessoa_fisica' tables. The second query selects data from 'pessoa' and 'pessoa_juridica' tables. The third query selects data from 'movimentacoes', 'produtos', 'pessoa_juridica', and 'pessoa' tables. The fourth query selects data from 'movimentacoes' table. The 'Data Output' tab is also visible, showing the results of the first query in a table format.

```
93
94 -- Consultas SQL conforme enunciado:
95
96 -- Dados completos de pessoas físicas
97 SELECT p.id_pessoa, p.nome, p.endereco, p.telefone, p.email, pf.cpf
98 FROM pessoa p
99 JOIN pessoa_fisica pf ON p.id_pessoa = pf.id_pessoa;
100
101 -- Dados completos de pessoas jurídicas
102 SELECT p.id_pessoa, p.nome, p.endereco, p.telefone, p.email, pj.cnpj
103 FROM pessoa p
104 JOIN pessoa_juridica pj ON p.id_pessoa = pj.id_pessoa;
105
106 -- Movimentações de entrada (compra):
107 -- . Produto
108 -- . Fornecedor (pessoa jurídica)
109 -- . Quantidade
110 -- . Preço unitário
111 -- . Valor total (quantidade x preço unitário)
112 SELECT
113     m.id_movimentacao,
114     pr.nome AS produto,
115     pe.nome AS fornecedor,
116     m.quantidade,
117     m.preco_unitario,
118     (m.quantidade * m.preco_unitario) AS valor_total
119 FROM movimentacoes m
120 JOIN produtos pr ON m.id_produto = pr.id_produto
121 JOIN pessoa_juridica pj ON pj.id_pessoa = pj.id_pessoa
122 JOIN pessoa pe ON pj.id_pessoa = pe.id_pessoa
123 WHERE m.tipo = 'E';
124
125 -- Movimentações de saída (venda):
126 -- .Produto
127 -- .Comprador (pessoa física)
128 -- .Quantidade
129 -- .Preço unitário
130 -- .Valor total
131 SELECT
132     m.id_movimentacao,
```

	id_pessoa integer	nome character varying (100)	endereco character varying (150)	telefone character varying (20)	email character varying (100)	cpf character varying (14)
1	1	João Silva	Rua das Flores, 123	(11) 98765-4321	joao@email.com	123.456.789-00

Segundo Procedimento:

Alimentando a Base

Códigos usados neste roteiro

The screenshot shows a SQL IDE interface with a query editor and a results pane. The query editor contains two SQL queries. The first query is a SELECT statement that joins the 'pessoa' table with the 'pessoa_juridica' table on the 'id_pessoa' column. The second query is a SELECT statement that joins the 'movimentacoes' table with the 'produtos' table on the 'id_produto' column, and also joins the 'pessoa_juridica' table on the 'id_pessoa' column. The results pane shows the output of the first query, which is a table with 7 columns: id_pessoa, nome, endereco, telefone, email, and cnpj. The results show one row with the following data: 2, Empresa XYZ Ltda., Av. Central, 5000, (11) 99876-5432, contato@empresa.com, 12.345.678/0001-99.

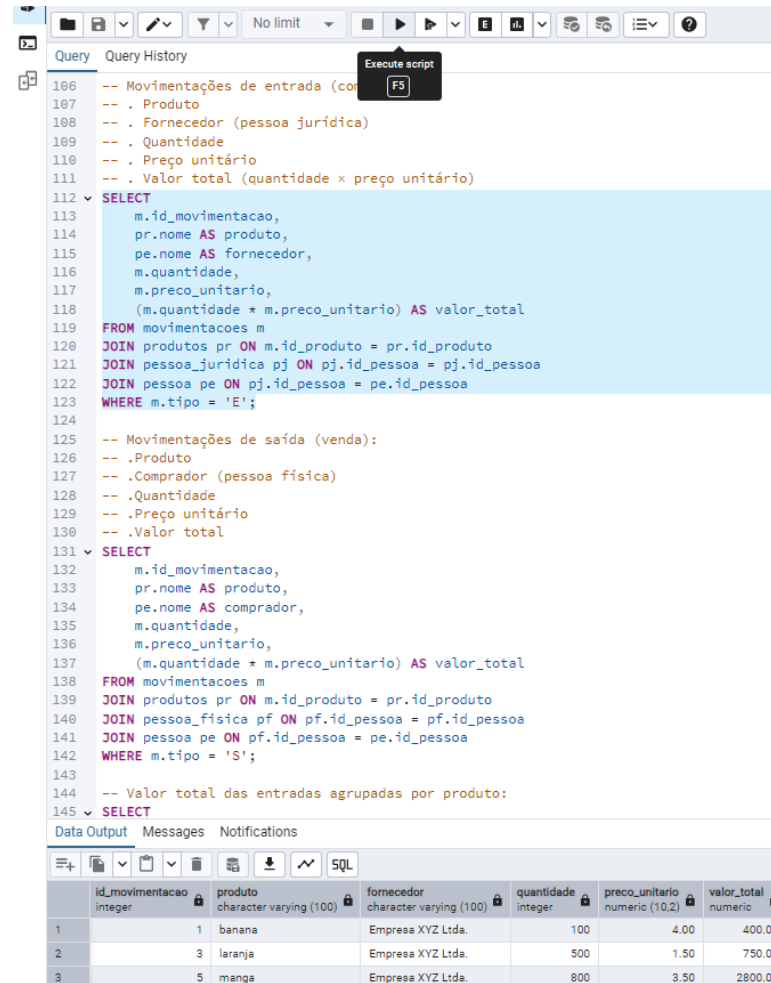
```
100
101 -- Dados completos de pessoas jurídicas
102 SELECT p.id_pessoa, p.nome, p.endereco, p.telefone, p.email, pj.cnpj
103 FROM pessoa p
104 JOIN pessoa_juridica pj ON p.id_pessoa = pj.id_pessoa;
105
106 -- Movimentações de entrada (compra):
107 -- . Produto
108 -- . Fornecedor (pessoa jurídica)
109 -- . Quantidade
110 -- . Preço unitário
111 -- . Valor total (quantidade x preço unitário)
112 SELECT
113     m.id_movimentacao,
114     pr.nome AS produto,
115     pe.nome AS fornecedor,
116     m.quantidade,
117     m.preco_unitario,
118     (m.quantidade * m.preco_unitario) AS valor_total
119 FROM movimentacoes m
120 JOIN produtos pr ON m.id_produto = pr.id_produto
121 JOIN pessoa_juridica pj ON pj.id_pessoa = p.id_pessoa
122 JOIN pessoa pe ON pj.id_pessoa = pe.id_pessoa
123 WHERE m.tipo = 'E';
124
125 -- Movimentações de saída (venda):
126 -- . Produto
127 -- . Comprador (pessoa física)
128 -- . Quantidade
129 -- . Preço unitário
130 -- . Valor total
131 SELECT
132     m.id_movimentacao,
133     pr.nome AS produto,
134     pe.nome AS comprador,
135     m.quantidade,
136     m.preco_unitario,
137     (m.quantidade * m.preco_unitario) AS valor_total
138 FROM movimentacoes m
139 JOIN produtos pr ON m.id_produto = pr.id_produto
```

	id_pessoa integer	nome character varying (100)	endereco character varying (150)	telefone character varying (20)	email character varying (100)	cnpj character varying (18)
1	2	Empresa XYZ Ltda.	Av. Central, 5000	(11) 99876-5432	contato@empresa.com	12.345.678/0001-99

Segundo Procedimento:

Alimentando a Base

Códigos usados neste roteiro



The screenshot shows a SQL IDE interface with a query editor and a results pane. The query editor contains two SQL queries. The first query is a SELECT statement that joins the 'movimentacoes' table with 'produtos', 'pessoa_juridica', and 'pessoa' tables. It filters for 'm.tipo = 'E'' and calculates the total value for each movement. The second query is a similar SELECT statement, but it filters for 'm.tipo = 'S'' and joins with 'pessoa_fisica' instead of 'pessoa_juridica'. The results pane shows the output of the first query, which is a table with 7 columns: id_movimentacao, produto, fornecedor, quantidade, preco_unitario, and valor_total. The results show 3 rows of data.

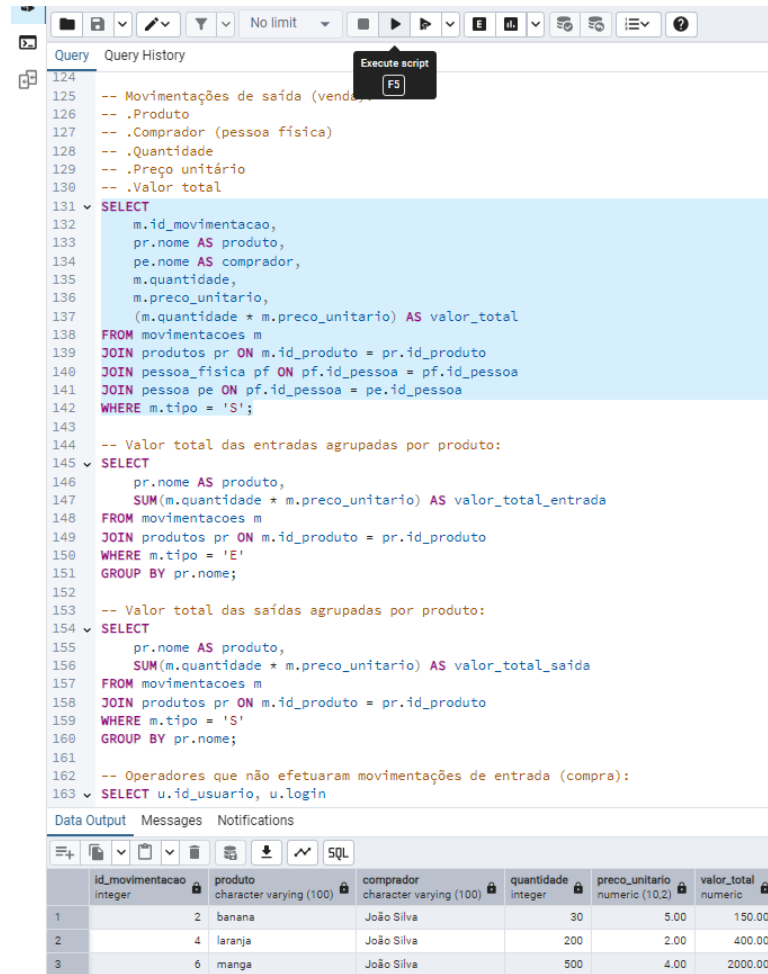
```
106 -- Movimentações de entrada (compra)
107 -- . Produto
108 -- . Fornecedor (pessoa jurídica)
109 -- . Quantidade
110 -- . Preço unitário
111 -- . Valor total (quantidade x preço unitário)
112 SELECT
113     m.id_movimentacao,
114     pr.nome AS produto,
115     pe.nome AS fornecedor,
116     m.quantidade,
117     m.preco_unitario,
118     (m.quantidade * m.preco_unitario) AS valor_total
119 FROM movimentacoes m
120 JOIN produtos pr ON m.id_produto = pr.id_produto
121 JOIN pessoa_juridica pj ON pj.id_pessoa = m.id_pessoa
122 JOIN pessoa pe ON pj.id_pessoa = pe.id_pessoa
123 WHERE m.tipo = 'E';
124
125 -- Movimentações de saída (venda):
126 -- . Produto
127 -- . Comprador (pessoa física)
128 -- . Quantidade
129 -- . Preço unitário
130 -- . Valor total
131 SELECT
132     m.id_movimentacao,
133     pr.nome AS produto,
134     pe.nome AS comprador,
135     m.quantidade,
136     m.preco_unitario,
137     (m.quantidade * m.preco_unitario) AS valor_total
138 FROM movimentacoes m
139 JOIN produtos pr ON m.id_produto = pr.id_produto
140 JOIN pessoa_fisica pf ON pf.id_pessoa = m.id_pessoa
141 JOIN pessoa pe ON pf.id_pessoa = pe.id_pessoa
142 WHERE m.tipo = 'S';
143
144 -- Valor total das entradas agrupadas por produto:
145 SELECT
```

id_movimentacao	produto	fornecedor	quantidade	preco_unitario	valor_total
1	1 banana	Empresa XYZ Ltda.	100	4.00	400.00
2	3 laranja	Empresa XYZ Ltda.	500	1.50	750.00
3	5 manga	Empresa XYZ Ltda.	800	3.50	2800.00

Segundo Procedimento:

Alimentando a Base

Códigos usados neste roteiro



```
124
125 -- Movimentações de saída (venda)
126 -- .Produto
127 -- .Comprador (pessoa física)
128 -- .Quantidade
129 -- .Preço unitário
130 -- .Valor total
131 SELECT
132     m.id_movimentacao,
133     pr.nome AS produto,
134     pe.nome AS comprador,
135     m.quantidade,
136     m.preco_unitario,
137     (m.quantidade * m.preco_unitario) AS valor_total
138 FROM movimentacoes m
139 JOIN produtos pr ON m.id_produto = pr.id_produto
140 JOIN pessoa_fisica pf ON pf.id_pessoa = m.id_pessoa
141 JOIN pessoa pe ON pe.id_pessoa = m.id_pessoa
142 WHERE m.tipo = 'S';
143
144 -- Valor total das entradas agrupadas por produto:
145 SELECT
146     pr.nome AS produto,
147     SUM(m.quantidade * m.preco_unitario) AS valor_total_entrada
148 FROM movimentacoes m
149 JOIN produtos pr ON m.id_produto = pr.id_produto
150 WHERE m.tipo = 'E'
151 GROUP BY pr.nome;
152
153 -- Valor total das saídas agrupadas por produto:
154 SELECT
155     pr.nome AS produto,
156     SUM(m.quantidade * m.preco_unitario) AS valor_total_saida
157 FROM movimentacoes m
158 JOIN produtos pr ON m.id_produto = pr.id_produto
159 WHERE m.tipo = 'S'
160 GROUP BY pr.nome;
161
162 -- Operadores que não efetuaram movimentações de entrada (compra):
163 SELECT u.id_usuario, u.login
```

	id_movimentacao integer	produto character varying (100)	comprador character varying (100)	quantidade integer	preco_unitario numeric (10,2)	valor_total numeric
1	2	banana	João Silva	30	5.00	150.00
2	4	laranja	João Silva	200	2.00	400.00
3	6	manga	João Silva	500	4.00	2000.00

Segundo Procedimento:

Alimentando a Base

Códigos usados neste roteiro



The screenshot shows a SQL IDE interface with a script editor and a results pane. The script contains four SQL queries. The first query calculates the total value of entries grouped by product. The second query calculates the total value of exits grouped by product. The third query identifies operators who have not performed entry movements. The fourth query calculates the total value of entries grouped by operator. The results pane shows the output of the first query.

```
143
144 -- Valor total das entradas agrupadas por produto:
145 SELECT
146     pr.nome AS produto,
147     SUM(m.quantidade * m.preco_unitario) AS valor_total_entrada
148 FROM movimentacoes m
149 JOIN produtos pr ON m.id_produto = pr.id_produto
150 WHERE m.tipo = 'E'
151 GROUP BY pr.nome;
152
153 -- Valor total das saídas agrupadas por produto:
154 SELECT
155     pr.nome AS produto,
156     SUM(m.quantidade * m.preco_unitario) AS valor_total_saida
157 FROM movimentacoes m
158 JOIN produtos pr ON m.id_produto = pr.id_produto
159 WHERE m.tipo = 'S'
160 GROUP BY pr.nome;
161
162 -- Operadores que não efetuaram movimentações de entrada (compra):
163 SELECT u.id_usuario, u.login
164 FROM usuarios u
165 WHERE u.id_usuario NOT IN (
166     SELECT DISTINCT id_usuario
167     FROM movimentacoes
168     WHERE tipo = 'E'
169 );
170
171 -- Valor total de entrada agrupado por operador:
172 SELECT
173     u.login,
174     SUM(m.quantidade * m.preco_unitario) AS valor_total_entrada
175 FROM movimentacoes m
176 JOIN usuarios u ON m.id_usuario = u.id_usuario
177 WHERE m.tipo = 'E'
178 GROUP BY u.login;
179
180 -- Valor total de saída agrupado por operador:
181 SELECT
182     u.login,
```

Data Output Messages Notifications

	produto character varying (100)	valor_total_entrada numeric
1	banana	400.00
2	laranja	750.00
3	manga	2800.00

Segundo Procedimento:

Alimentando a Base

Códigos usados neste roteiro



The screenshot shows a SQL IDE interface with a script editor and a results pane. The script contains four SQL queries, each preceded by a comment. The first query calculates the total value of sales grouped by product. The second query identifies operators who have not performed any entry movements. The third query calculates the total value of entry movements grouped by operator. The fourth query calculates the total value of sales grouped by operator. The results pane shows the output of the first query, displaying a table with two columns: 'produto' and 'valor_total_saida'.

```
152
153 -- Valor total das saídas agrupadas por produto:
154 SELECT
155     pr.nome AS produto,
156     SUM(m.quantidade * m.preco_unitario) AS valor_total_saida
157 FROM movimentacoes m
158 JOIN produtos pr ON m.id_produto = pr.id_produto
159 WHERE m.tipo = 'S'
160 GROUP BY pr.nome;
161
162 -- Operadores que não efetuaram movimentações de entrada (compra):
163 SELECT u.id_usuario, u.login
164 FROM usuarios u
165 WHERE u.id_usuario NOT IN (
166     SELECT DISTINCT id_usuario
167     FROM movimentacoes
168     WHERE tipo = 'E'
169 );
170
171 -- Valor total de entrada agrupado por operador:
172 SELECT
173     u.login,
174     SUM(m.quantidade * m.preco_unitario) AS valor_total_entrada
175 FROM movimentacoes m
176 JOIN usuarios u ON m.id_usuario = u.id_usuario
177 WHERE m.tipo = 'E'
178 GROUP BY u.login;
179
180 -- Valor total de saída agrupado por operador:
181 SELECT
182     u.login,
183     SUM(m.quantidade * m.preco_unitario) AS valor_total_saida
184 FROM movimentacoes m
185 JOIN usuarios u ON m.id_usuario = u.id_usuario
186 WHERE m.tipo = 'S'
187 GROUP BY u.login;
188
189 -- Valor médio de venda por produto (média ponderada):
190 SELECT
191     pr.nome AS produto,
```

	produto character varying (100)	valor_total_saida numeric
1	banana	150.00
2	laranja	400.00
3	manga	2000.00

Segundo Procedimento:

Alimentando a Base

Códigos usados neste roteiro

```
161
162 -- Operadores que não efetuaram movimentações de entrada (compra):
163 SELECT u.id_usuario, u.login
164 FROM usuarios u
165 WHERE u.id_usuario NOT IN (
166     SELECT DISTINCT id_usuario
167     FROM movimentacoes
168     WHERE tipo = 'E'
169 );
170
171 -- Valor total de entrada agrupado por operador:
172 SELECT
173     u.login,
174     SUM(m.quantidade * m.preco_unitario) AS valor_total_entrada
175 FROM movimentacoes m
176 JOIN usuarios u ON m.id_usuario = u.id_usuario
177 WHERE m.tipo = 'E'
178 GROUP BY u.login;
179
180 -- Valor total de saída agrupado por operador:
181 SELECT
182     u.login,
183     SUM(m.quantidade * m.preco_unitario) AS valor_total_saida
184 FROM movimentacoes m
185 JOIN usuarios u ON m.id_usuario = u.id_usuario
186 WHERE m.tipo = 'S'
187 GROUP BY u.login;
188
189 -- Valor médio de venda por produto (média ponderada):
190 SELECT
191     pr.nome AS produto,
192     SUM(m.quantidade * m.preco_unitario) / SUM(m.quantidade) AS media_ponderada
193 FROM movimentacoes m
194 JOIN produtos pr ON m.id_produto = pr.id_produto
195 WHERE m.tipo = 'S'
196 GROUP BY pr.nome;
197
198
199
200
```

Data Output Messages Notifications

	id_usuario [PK] integer	login character varying (50)
1	2	op2

Segundo Procedimento:

Alimentando a Base

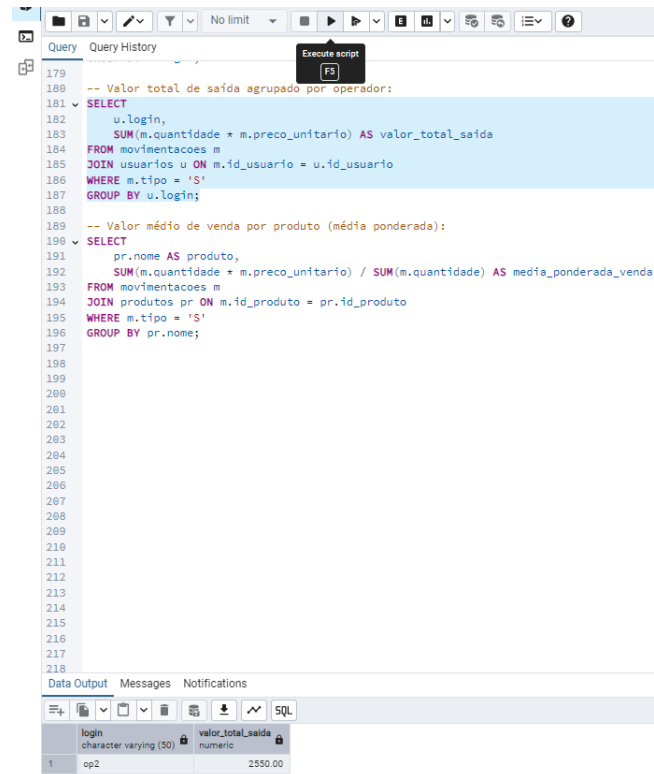
Códigos usados neste roteiro

```
Query Query History
-- Valor total de entrada agrupado por operador:
171 SELECT
172     u.login,
173     SUM(m.quantidade * m.preco_unitario) AS valor_total_entrada
174 FROM movimentacoes m
175 JOIN usuarios u ON m.id_usuario = u.id_usuario
176 WHERE m.tipo = 'E'
177 GROUP BY u.login;
178
179
180 -- Valor total de saída agrupado por operador:
181 SELECT
182     u.login,
183     SUM(m.quantidade * m.preco_unitario) AS valor_total_saida
184 FROM movimentacoes m
185 JOIN usuarios u ON m.id_usuario = u.id_usuario
186 WHERE m.tipo = 'S'
187 GROUP BY u.login;
188
189 -- Valor médio de venda por produto (média ponderada):
190 SELECT
191     pr.nome AS produto,
192     SUM(m.quantidade * m.preco_unitario) / SUM(m.quantidade) AS media_ponderada_venda
193 FROM movimentacoes m
194 JOIN produtos pr ON m.id_produto = pr.id_produto
195 WHERE m.tipo = 'S'
196 GROUP BY pr.nome;
197
198
199
200
201
202
203
204
205
206
207
208
209
210
Data Output Messages Notifications
login valor_total_entrada
character varying (50) numeric
1 op1 3950.00
```

Segundo Procedimento:

Alimentando a Base

Códigos usados neste roteiro



The screenshot shows a SQL IDE interface with a query editor and a results pane. The query editor contains two SQL queries. The first query calculates the total value of sales grouped by operator. The second query calculates the average value of sales per product (weighted average). The results pane shows the output of the first query, which is a table with two columns: 'login' and 'valor_total_saida'.

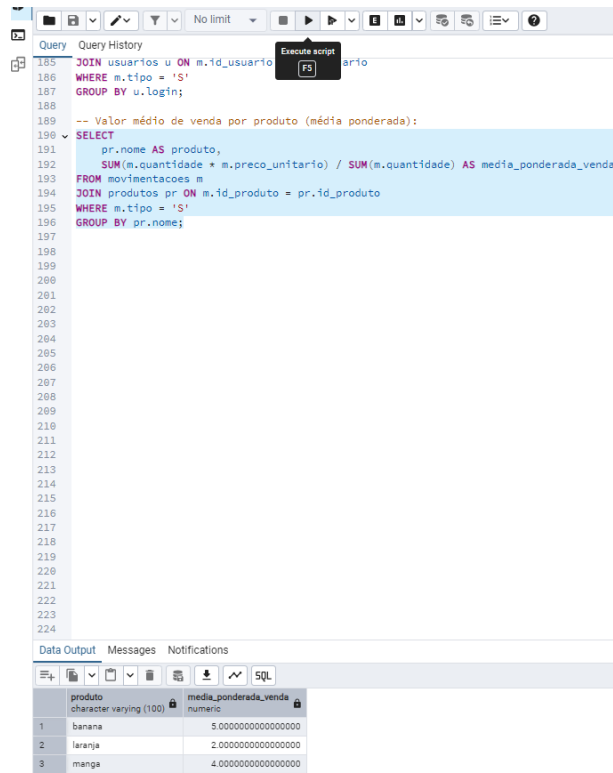
```
179 -- Valor total de saída agrupado por operador:
180
181 SELECT
182     u.login,
183     SUM(m.quantidade * m.preco_unitario) AS valor_total_saida
184 FROM movimentacoes m
185 JOIN usuarios u ON m.id_usuario = u.id_usuario
186 WHERE m.tipo = 'S'
187 GROUP BY u.login;
188
189 -- Valor médio de venda por produto (média ponderada):
190 SELECT
191     pr.nome AS produto,
192     SUM(m.quantidade * m.preco_unitario) / SUM(m.quantidade) AS media_ponderada_venda
193 FROM movimentacoes m
194 JOIN produtos pr ON m.id_produto = pr.id_produto
195 WHERE m.tipo = 'S'
196 GROUP BY pr.nome;
```

login	valor_total_saida
op2	2550.00

Segundo Procedimento:

Alimentando a Base

Códigos usados neste roteiro



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, query execution, and settings. The main editor displays a SQL query with line numbers 185 to 224. The query is as follows:

```
185 JOIN usuarios u ON m.id_usuario = u.id_usuario
186 WHERE m.tipo = 'S'
187 GROUP BY u.login;
188
189 -- Valor médio de venda por produto (média ponderada):
190 SELECT
191     pr.nome AS produto,
192     SUM(m.quantidade * m.preco_unitario) / SUM(m.quantidade) AS media_ponderada_venda
193 FROM movimentacoes m
194 JOIN produtos pr ON m.id_produto = pr.id_produto
195 WHERE m.tipo = 'S'
196 GROUP BY pr.nome;
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
```

Below the query editor, there is a 'Data Output' tab. It shows the results of the query in a table with two columns: 'produto' (character varying (100)) and 'media_ponderada_venda' (numeric). The results are as follows:

	produto	media_ponderada_venda
1	banana	5.0000000000000000
2	laranja	2.0000000000000000
3	manga	4.0000000000000000

Análise e conclusão:

Quais as diferenças no uso de sequence e identity?

As diferenças entre o uso de SEQUENCE e IDENTITY no SQL Server estão relacionadas à forma como os valores únicos são gerados para identificadores (como chaves primárias). Ambos os recursos têm como objetivo fornecer valores incrementais automaticamente, mas eles funcionam de maneiras diferentes e são adequados para cenários distintos.

Qual a importância das chaves estrangeiras para a consistência do banco?

As chaves estrangeiras (ou foreign keys) desempenham um papel fundamental na garantia da consistência e integridade dos dados em um banco de dados relacional. Elas estabelecem relacionamentos entre tabelas, assegurando que os dados armazenados sejam válidos e consistentes com as regras de negócio definidas.

Quais operadores do SQL pertencem à álgebra relacional e quais são definidos no cálculo relacional?

A álgebra relacional e o cálculo relacional são duas abordagens formais para consultar e manipular dados em bancos de dados relacionais. Embora ambos sejam equivalentes em termos de poder expressivo (ou seja, qualquer consulta que pode ser escrita em uma pode ser traduzida para a outra), eles diferem na forma como representam as operações.

Análise e conclusão:

Como é feito o agrupamento em consultas, e qual requisito é obrigatório?

O agrupamento em consultas SQL é realizado usando a cláusula GROUP BY. Ele é usado para organizar linhas de uma tabela em grupos com base nos valores de uma ou mais colunas. O agrupamento é especialmente útil quando você deseja aplicar funções de agregação (como COUNT, SUM, AVG, MIN, MAX) a cada grupo separadamente.

LINK DO GITHUB:

https://github.com/mauriciocampos1234/Missao_Pratica_2_BackEnd_Estacio