
MATERIAL - BANCOS DE DADOS

LINGUAGEM SQL
DQL (Data Query Language)



BANCOS DE DADOS

- ❖ CONTEÚDO PROGRAMÁTICO:

- ❖ 2.2 - Funções da linguagem SQL.

- ❖ 2.2.1 - Funções agregadoras.

- ❖ 2.2.2 - Funções escalares.

BANCOS DE DADOS

- ❖ CONTEÚDO PROGRAMÁTICO:
 - ❖ 3.1 Operações de consultas de dados (DQL);
 - ❖ 3.1.1 Clausulas SQL;
 - ❖ 3.1.2 Select utilizando operações aritméticas e matemáticas;
 - ❖ 3.1.3 Operadores de concatenação, Cadeia de caracteres (String);
 - ❖ 3.1.5 Manuseio de valores nulos;

BANCOS DE DADOS

❖ 1.1 - Conceitos de Bancos de dados e SQL;

A linguagem padrão de bancos de dados relacionais é SQL que é dividida em 5 grupos:

- 1 - DDL - Linguagem de Definição de Dados.
- 2 - DML - Linguagem de Manipulação de Dados.
- 3 - DCL - Linguagem de Controle de Dados.
- 4 - DQL - Linguagem de Consulta de Dados.
- 5 - DTL - Linguagem de Transação de Dados.

BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.

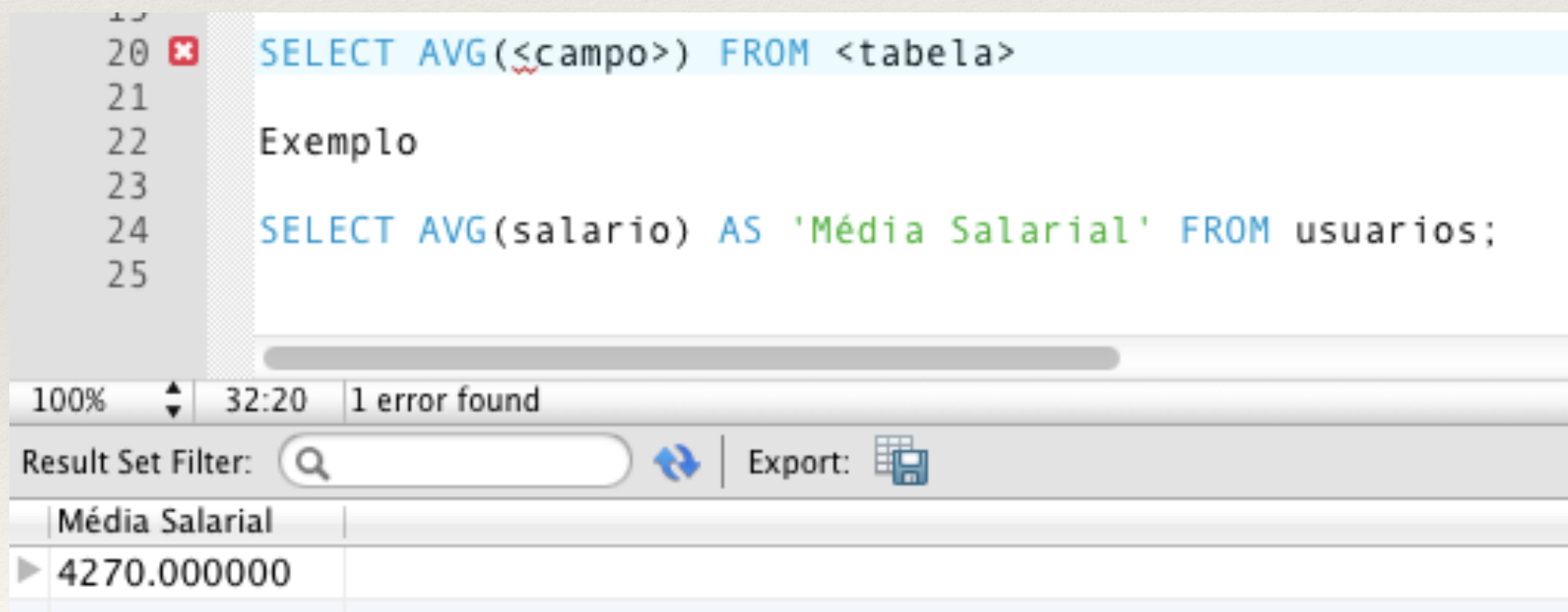
- Funções agregadoras:

São funções que retornam um valor calculado a partir dos dados da(s) coluna(s) referenciadas. As principais funções são, **AVG, COUNT, MAX, MIN, SUM, ORDER BY, DISTINCT, GROUP BY, HAVING.**

BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.

- **Função AVG:** Esta função retorna a média aritmética de um campo numérico.



The screenshot shows a SQL IDE interface. On the left, a line editor shows line numbers 19 to 25. Line 20 contains the syntax `SELECT AVG(<campo>) FROM <tabela>`. Line 22 contains the word "Exemplo". Line 24 contains the example query `SELECT AVG(salario) AS 'Média Salarial' FROM usuarios;`. The bottom of the IDE shows a status bar with "100%", "32:20", and "1 error found". Below the status bar is a "Result Set Filter" search bar and an "Export" button. The result set table has two columns: "Média Salarial" and a value "4270.000000".

```
SELECT AVG(<campo>) FROM <tabela>
```

Exemplo

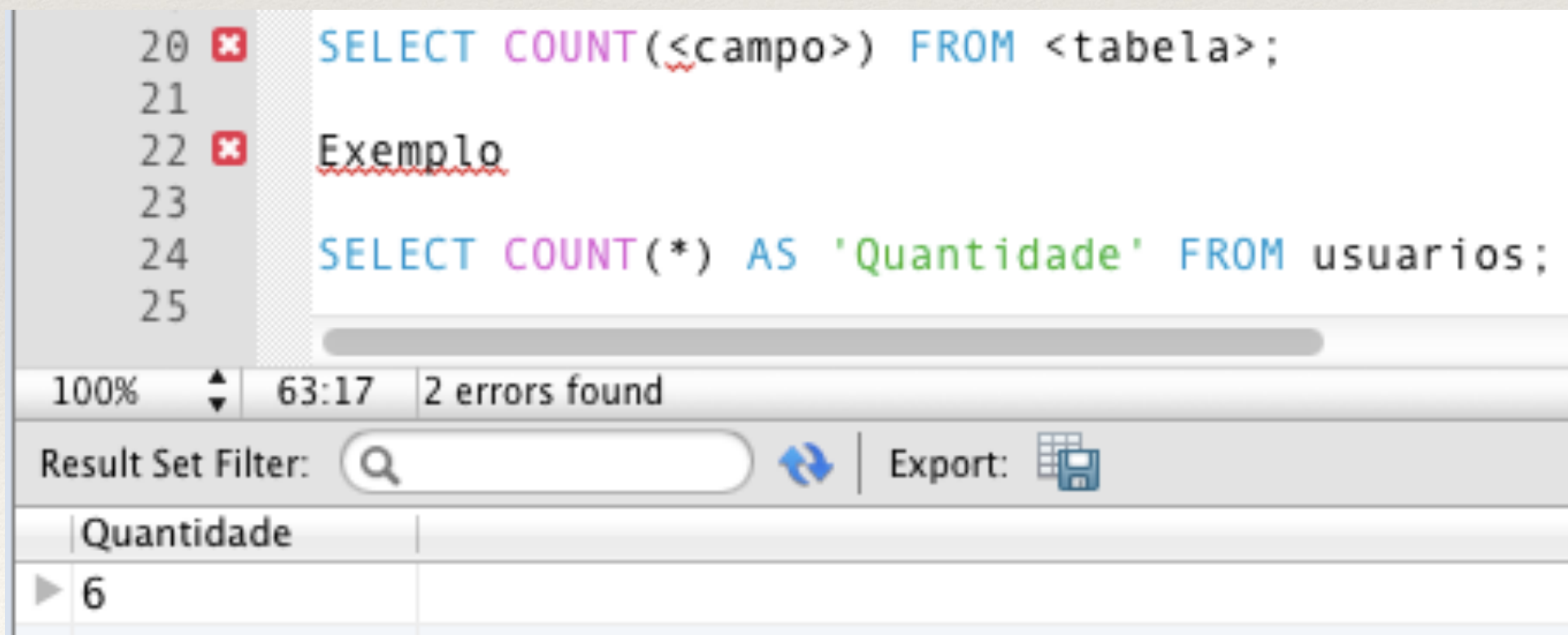
```
SELECT AVG(salario) AS 'Média Salarial' FROM usuarios;
```

Média Salarial
4270.000000

BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.


- **Função COUNT:** Esta função retorna a quantidade de registros que combinam com um determinado critério de seleção.



The screenshot shows a SQL IDE interface. The editor displays two SQL queries. The first query is `SELECT COUNT(<campo>) FROM <tabela>;`. The second query is `SELECT COUNT(*) AS 'Quantidade' FROM usuarios;`. Below the editor, the status bar indicates '100%' zoom, '63:17' cursor position, and '2 errors found'. The 'Result Set Filter' is empty. The 'Export' button is visible. The result set shows a single row with the value '6' under the column 'Quantidade'.

```
20 ✖ SELECT COUNT(<campo>) FROM <tabela>;
21
22 ✖ Exemplo
23
24 SELECT COUNT(*) AS 'Quantidade' FROM usuarios;
25
```

100% 63:17 2 errors found

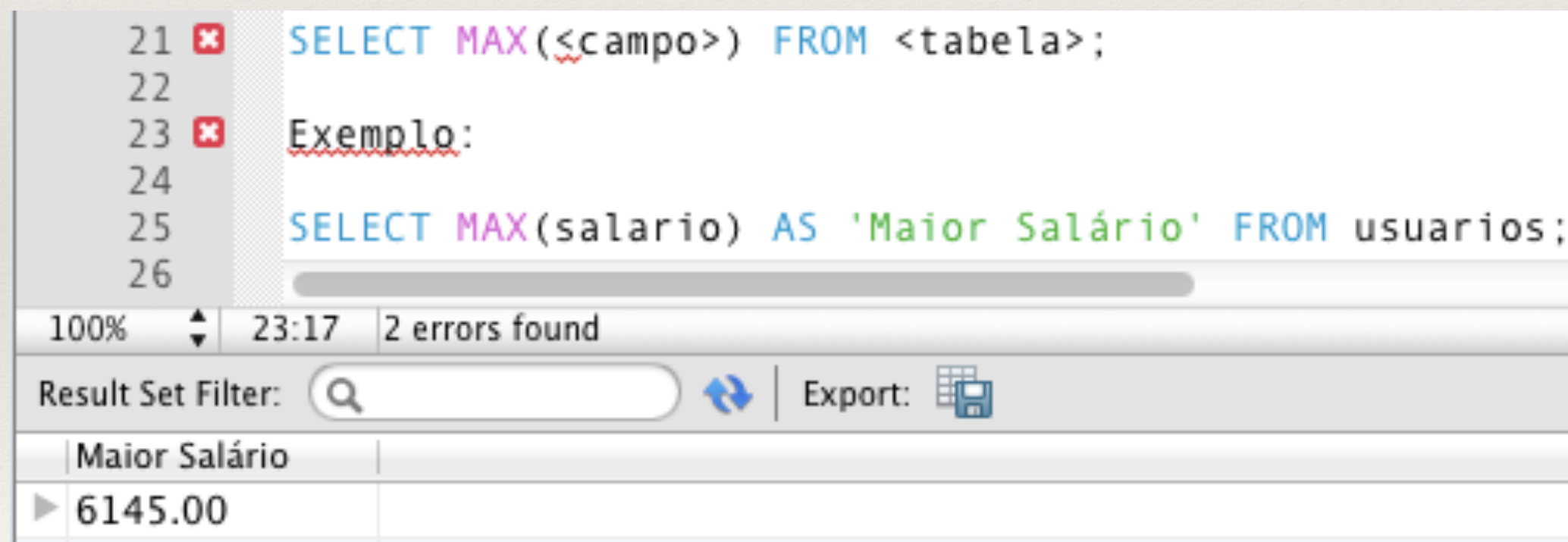
Result Set Filter: Export: 

Quantidade
6

BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.

- **Função MAX:** Esta função retorna o maior valor de um determinado campo selecionado.



The screenshot shows a SQL IDE interface. The query editor contains the following SQL code:

```
21 SELECT MAX(<campo>) FROM <tabela>;  
22  
23 Exemplo:  
24  
25 SELECT MAX(salario) AS 'Maior Salário' FROM usuarios;  
26
```

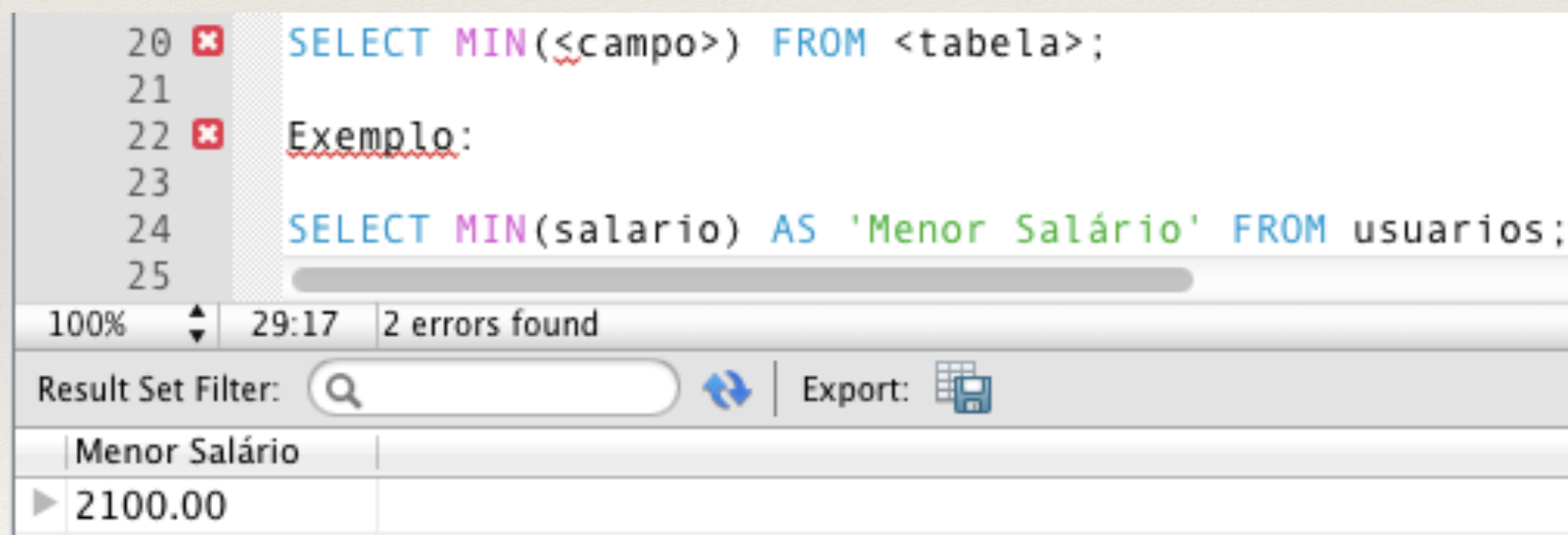
Below the query editor, the status bar indicates "100%", "23:17", and "2 errors found". The "Result Set Filter" is empty. The "Export" button is visible. The result set is displayed in a table with one column, "Maior Salário", and one row with the value "6145.00".

Maior Salário
6145.00



BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.

- **Função MIN:** Esta função retorna o menor valor de um determinado campo selecionado.



The screenshot shows a SQL IDE interface. The query editor contains the following text:

```
20  SELECT MIN(<campo>) FROM <tabela>;  
21  
22  Exemplo:  
23  
24 SELECT MIN(salario) AS 'Menor Salário' FROM usuarios;  
25
```

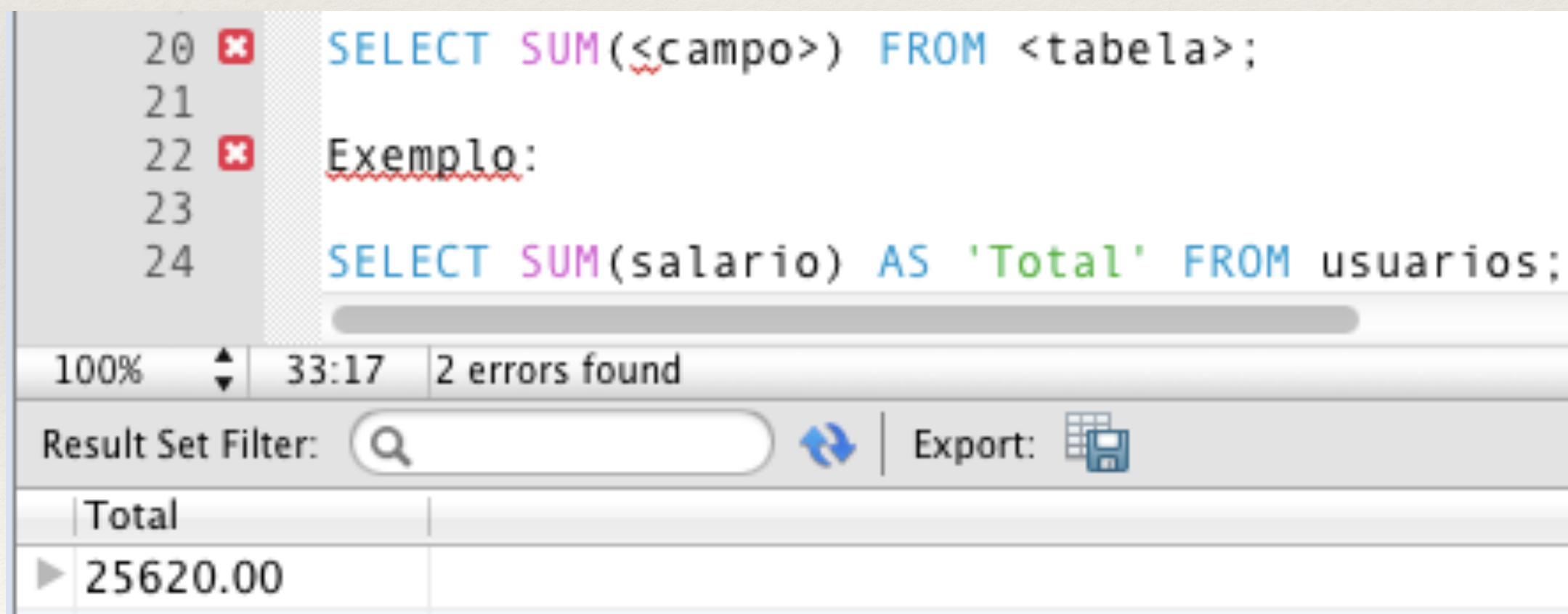
Below the editor, a status bar shows "100%", a zoom icon, "29:17", and "2 errors found". Below that is a "Result Set Filter" with a search icon and an "Export" button with a document icon. The results pane shows a single row with the column name "Menor Salário" and the value "2100.00".

Menor Salário
2100.00



BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.

- **Função SUM:** Esta função retorna a soma total de um campo numérico.



The screenshot shows a SQL IDE interface. The editor contains the following SQL code:

```
20  SELECT SUM(<campo>) FROM <tabela>;  
21  
22  Exemplo:  
23  
24 SELECT SUM(salario) AS 'Total' FROM usuarios;
```


Below the editor, the status bar indicates "100%", "33:17", and "2 errors found". The "Result Set Filter" is empty. The "Export" button is visible. The result set is displayed in a table with one row:

Total
25620.00



BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.

- **Função ORDER BY:** Utilizada ordenar o campos selecionado em ordem crescente(ASC) ou decrescente(DESC).

```
20  SELECT <campo/valor agregado> FROM <tabela>
21     WHERE <condição> ORDER BY <campo a ser ordenado>
22     <formato da ordenação>
23
24 Exemplo:
25
26 SELECT data_admissao FROM usuarios
27     WHERE salario>3000 Order BY data_admissao ASC;
```

100% 61:17 1 error found

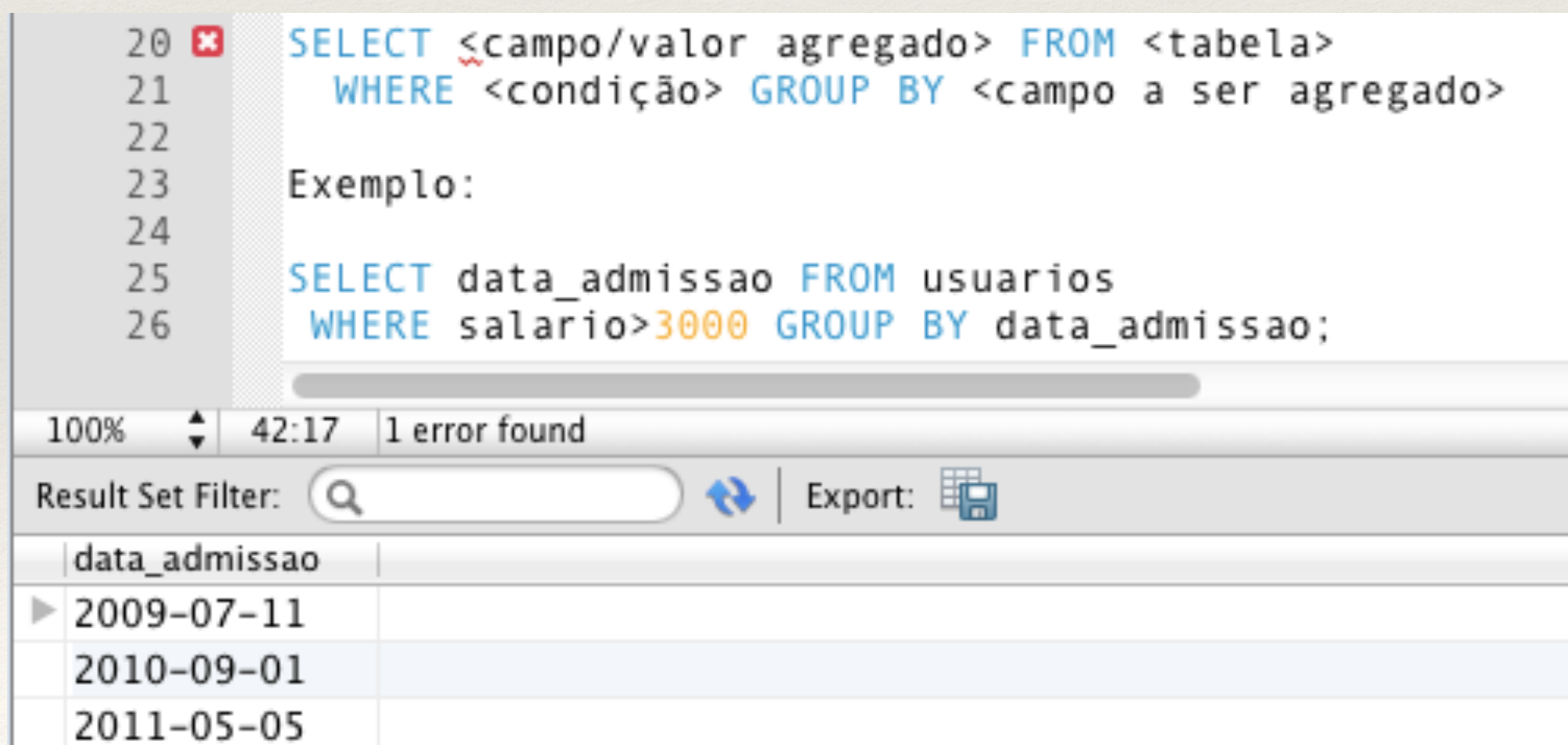
Result Set Filter:  Export: 

	data_admissao
▶	2009-07-11
	2009-07-11
	2010-09-01

BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.

- **Função GROUP BY:** Utilizada para agrupar registros (agregar) com campos semelhantes.



The screenshot shows a SQL IDE interface. The editor displays a SQL query template and an example. The query template is: `SELECT <campo/valor agregado> FROM <tabela> WHERE <condição> GROUP BY <campo a ser agregado>`. The example query is: `SELECT data_admissao FROM usuarios WHERE salario > 3000 GROUP BY data_admissao;`. Below the editor, the status bar shows '100%' zoom, '42:17' cursor position, and '1 error found'. The 'Result Set Filter' is empty. The 'Export' button is visible. The results table has a header 'data_admissao' and three rows of dates: '2009-07-11', '2010-09-01', and '2011-05-05'.

```
20 SELECT <campo/valor agregado> FROM <tabela>
21     WHERE <condição> GROUP BY <campo a ser agregado>
22
23 Exemplo:
24
25 SELECT data_admissao FROM usuarios
26     WHERE salario > 3000 GROUP BY data_admissao;
```

100% 42:17 1 error found

Result Set Filter: Export:

data_admissao
2009-07-11
2010-09-01
2011-05-05

BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.

- **Função HAVING:** A cláusula HAVING é utilizada em conjunto com a cláusula GROUP BY. É a expressão condicional para campos e valores agregados, pois a cláusula WHERE trabalha somente com valores não agregados (simples).

```
21
22 ❌ SELECT <campo/valor agregado> FROM <tabela>
23     WHERE <condição> GROUP BY <campo a ser agregado>
24     HAVING <condição agregada>
25
26 Exemplos:
27
28 SELECT salario AS 'Total' FROM usuarios
29     GROUP BY salario HAVING AVG(salario)>3000;
```

100% 3:13 1 error found

Result Set Filter: Export:

Total
3500.00
4800.00
5575.00
6145.00

BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.

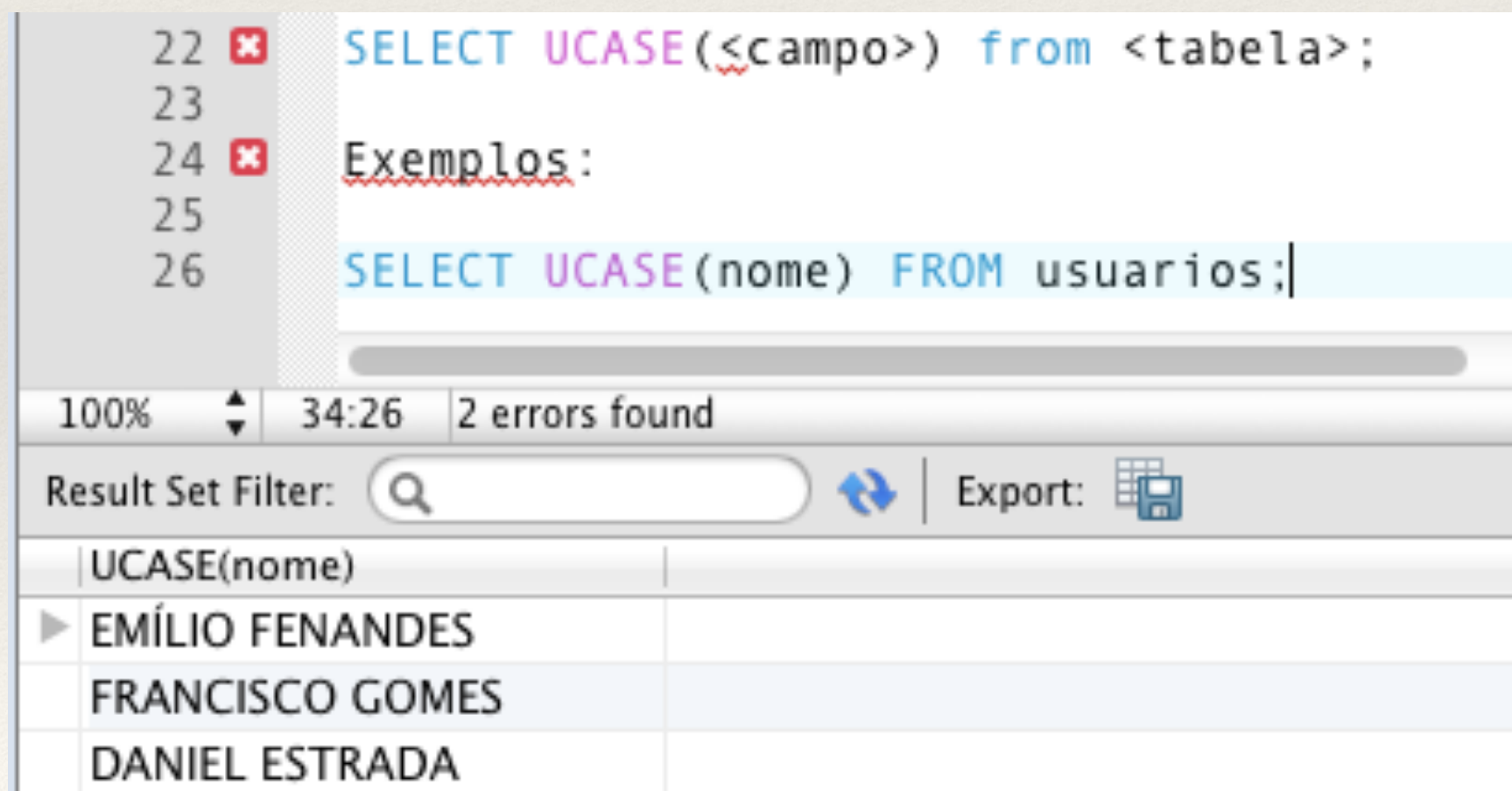
- Funções escalares:

São funções que retornam um valor calculado a partir de um valor de entrada. As principais funções são, **UCASE**, **LCASE**, **MID**, **LENGTH**, **ROUND**, **NOW**, **FORMAT**.

BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.

- **UCASE ()** - Retorna o valor dos campos em letras maiúsculas.



The screenshot shows a SQL IDE interface. The editor displays the following SQL code:

```
22 SELECT UCASE(<campo>) from <tabela>;  
23  
24 Exemplos:  
25  
26 SELECT UCASE(nome) FROM usuarios;
```

Below the editor, the status bar indicates "100%", "34:26", and "2 errors found". The "Result Set Filter" is empty. The "Export" button is visible. The results table shows the output of the UCASE function applied to the 'nome' column of the 'usuarios' table.

UCASE(nome)
EMÍLIO FENANDES
FRANCISCO GOMES
DANIEL ESTRADA

BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.

- **LCASE ()** - Retorna o valor dos campos em letras minúsculas.

The screenshot shows a SQL IDE interface. The editor displays the following SQL code:

```
22 SELECT LCASE(<campo>) from <tabela>;  
23  
24 Exemplos:  
25  
26 SELECT LCASE(nome) FROM usuarios;  
27  
28
```

Below the editor, the status bar indicates "100%", "41:17", and "2 errors found". The "Result Set Filter" is empty. The "Export" button is visible. The results table shows the output of the LCASE function applied to the 'nome' column of the 'usuarios' table.

LCASE(nome)
emílio fenandes
francisco gomes
daniel estrada
célio drummond
abelardo barb...
bernardo costa

BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.

- **MID ()** - é utilizado para extrair caracteres a partir de um campo de texto .

```
21
22 ✖ SELECT MID(<campo>, inicio, fim) from <tabela>;
23
24 ✖ Exemplo:
25
26 ● SELECT MID(nome, 1, 8) FROM usuarios;
27
28
```

100% 25:22 2 errors found

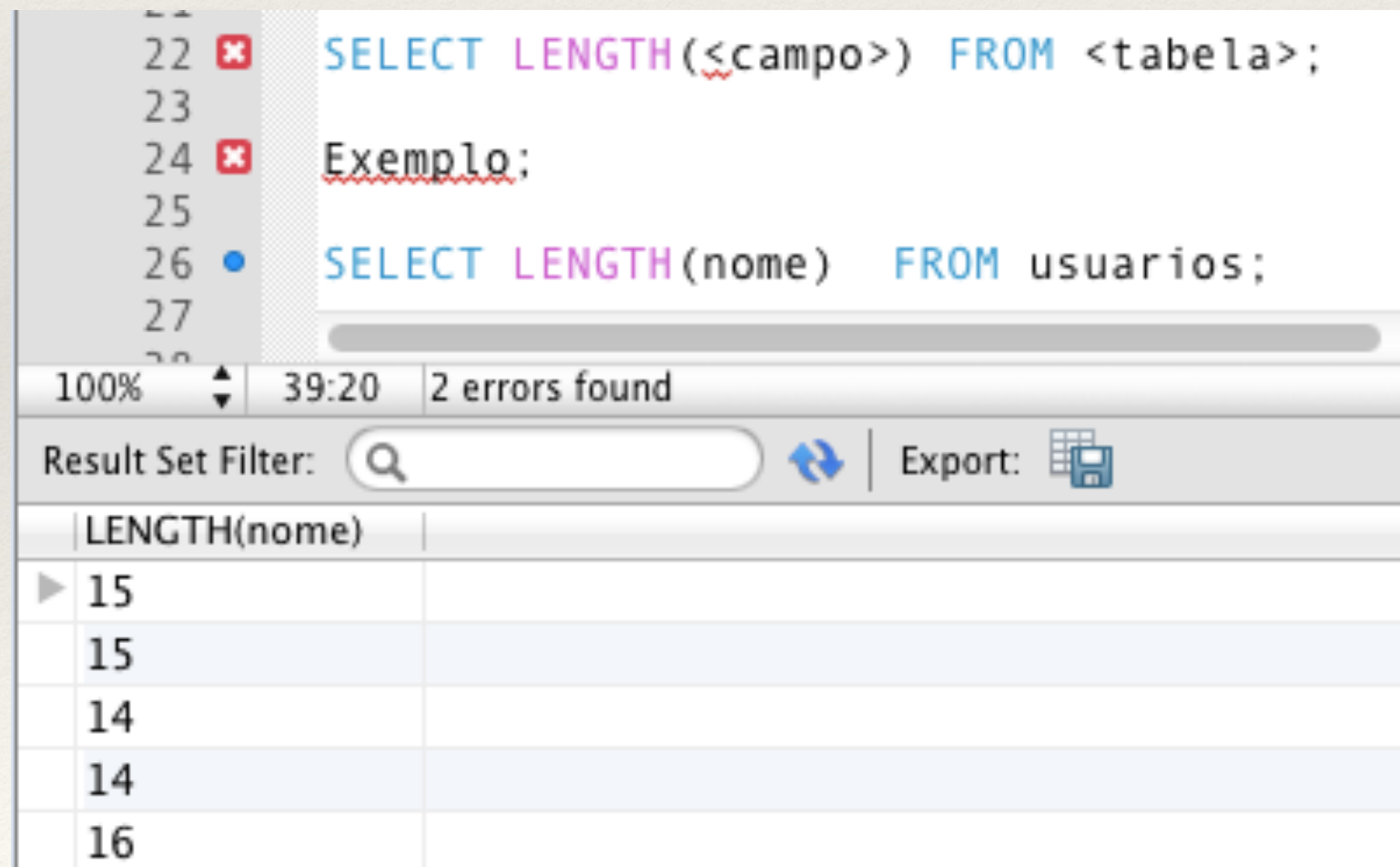
Result Set Filter: 🔍 | Export: 📄

MID(nome,1,8)
▶ Emílio F
Francisc
Daniel E

BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.

- **LENGTH ()** - retorna o comprimento do valor de um campo.



The screenshot shows a SQL IDE interface. The query editor displays the following SQL code:

```
22 SELECT LENGTH(<campo>) FROM <tabela>;  
23  
24 Exemplo;  
25  
26 SELECT LENGTH(nome) FROM usuarios;  
27
```

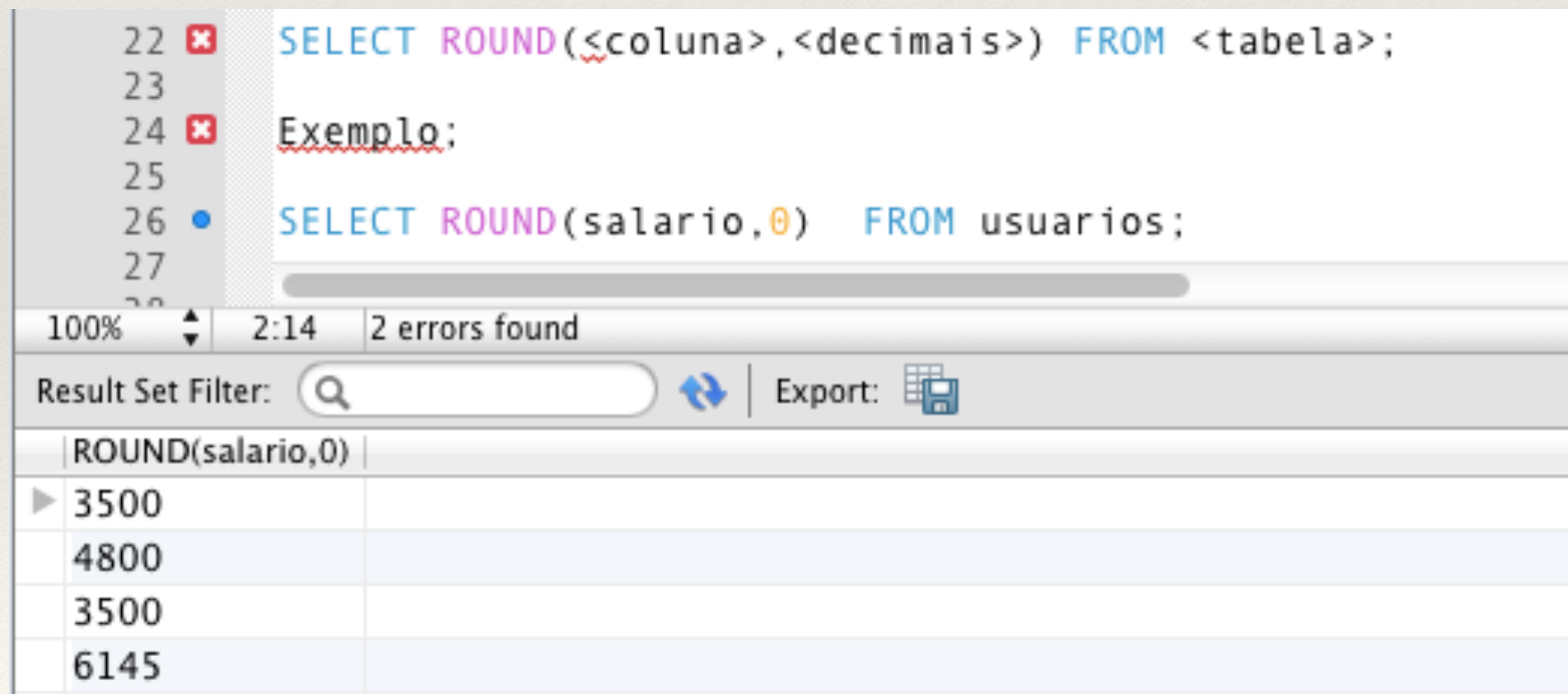
Below the query editor, the status bar indicates "100%", "39:20", and "2 errors found". The "Result Set Filter" is set to "Q". The "Export" button is visible. The results table shows the output of the query:

LENGTH(nome)
15
15
14
14
16

BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.

- **ROUND()** - é usado para arredondar um campo numérico para o número de casas decimais especificado.



The screenshot shows a SQL IDE interface. The editor displays the following SQL code:

```
22 ✖ SELECT ROUND(<coluna>,<decimais>) FROM <tabela>;
23
24 ✖ Exemplo:
25
26 ● SELECT ROUND(salario,0) FROM usuarios;
27
```

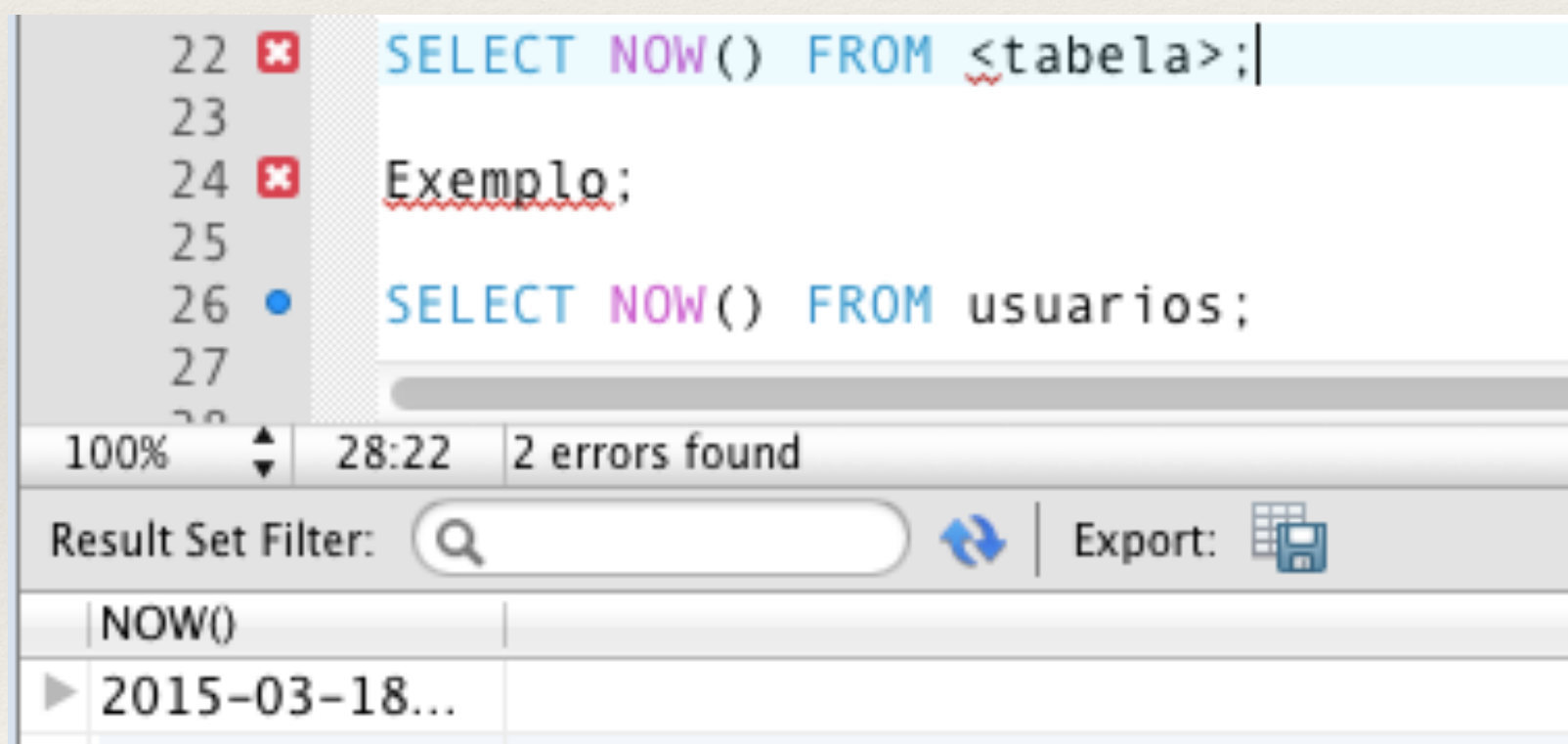
Below the editor, the status bar indicates "100%", "2:14", and "2 errors found". The "Result Set Filter" is empty. The "Export" button is visible. The result set is displayed in a table with the following data:

ROUND(salario,0)
3500
4800
3500
6145

BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.

- **NOW()** - retorna a data atual do sistema e tempo.



The screenshot shows a SQL IDE interface. The editor window contains the following text:

```
22 ✖ SELECT NOW() FROM <tabela>;|
23
24 ✖ Exemplo;
25
26 ● SELECT NOW() FROM usuarios;
27
```

Below the editor, the status bar indicates "100%", "28:22", and "2 errors found". The "Result Set Filter" section includes a search input and an "Export" button. The results pane shows a single row with the value "2015-03-18..." under the column header "NOW()".

NOW()
2015-03-18...

BANCOS DE DADOS

❖ 2.2 - Funções da linguagem SQL.

- **FORMAT()** - é usada para formatar como um campo é deverá ser exibido .

```
22 ✖ SELECT FORMAT(<coluna>,<formatação>) FROM <tabela>;
23
24 ✖ Exemplo:
25
26 ● SELECT nome, FORMAT(salario,1) FROM usuarios;
27
```

100% 54:18 2 errors found

Result Set Filter: Export:

nome	FORMAT(salario,1)
Emílio Fenandes	3,500.0
Francisco Gomes	4,800.0
Daniel Estrada	3,500.0

BANCOS DE DADOS

❖ 3.1 Operações de consultas de dados (DQL):

Clausula **WHERE**:

A cláusula **WHERE** é usado para filtrar registros.

```
SELECT <coluna> FROM <tabela>  
WHERE <coluna><operator><valor>;
```

```
SELECT NumTitulo FROM Eleitor  
WHERE NumTitulo=125;
```

BANCOS DE DADOS

❖ 3.1 Operações de consultas de dados (DQL):

Clausula **DISTINCT**:

A instrução **SELECT DISTINCT** é utilizada para retornar apenas valores distintos (diferentes) .

```
select DISTINCT <coluna> from <tabela>;
```

```
select DISTINCT ZONA_ELEITORAL_NumCidade from eleitor;
```


BANCOS DE DADOS

❖ 3.1 Operações de consultas de dados (DQL):

Clausula LIKE:

O operador LIKE é utilizado em uma cláusula WHERE para pesquisar um valor especificado em uma coluna.

```
SELECT <coluna> FROM <tabela>  
WHERE <coluna> LIKE <valor>;
```

```
SELECT NomeEleitor FROM ELEITOR  
where NomeEleitor LIKE 'Carla';|
```

BANCOS DE DADOS

❖ 3.1 Operações de consultas de dados (DQL):

Símbolos **CORINGA**:

Em SQL, caracteres curinga são utilizados com o operador SQL LIKE. Wildcards SQL são utilizados para pesquisar os dados em uma tabela.

Símbolo	Descrição
%	Um substituto para zero ou mais caracteres.
—	Um substituto para um único caracter.

BANCOS DE DADOS

❖ 3.1 Operações de consultas de dados (DQL):

Usando o SQL %:

A seguinte instrução SQL selecionar todos os clientes com uma cidade começando com "C":

```
SELECT <coluna> FROM <tabela>  
WHERE <coluna> LIKE <valor>;  
  
SELECT NomeEleitor FROM ELEITOR  
where NomeEleitor LIKE 'C%';
```

BANCOS DE DADOS

❖ 3.1 Operações de consultas de dados (DQL):

Usando o SQL _:

A seguinte instrução SQL selecionar todos os nomes com qualquer caracter seguido por "Arla":

```
SELECT NomeEleitor FROM Eleitor  
WHERE NomeEleitor LIKE '_arla';
```

BANCOS DE DADOS

❖ 3.1 Operações de consultas de dados (DQL):

Usando o SQL com operações aritméticas:

```
/* Adição: */  
select salario + id_usuario from usuarios;  
  
/* Subtração: */  
select salario - id_usuario from usuarios;  
  
/* Multiplicação: */  
select salario * id_usuario from usuarios;  
  
/* Divisão: */  
select salario / id_usuario from usuarios;
```

BANCOS DE DADOS

❖ 3.1 Operações de consultas de dados (DQL):

Usando o SQL com operações matemáticas:

```
/* ABS(X) - Retorna o valor absoluto */
```

```
SELECT ABS(salario) from usuarios;
```

```
/* SIGN(X) - Retorna o sinal do argumento como -1, 0, ou 1,  
dependendo de quando X é negativo, zero, ou positivo */
```

```
SELECT SIGN(salario) from usuarios;
```

```
/* MOD(N,M) - Retorna o resto de N dividido por M */
```

```
SELECT salario, mod(salario, id_usuario)  
from usuarios;
```


BANCOS DE DADOS

❖ 3.1 Operações de consultas de dados (DQL):

Usando o SQL com operações matemáticas:

```
/* POWER(X,Y) - Retorna o valor X na potência Y */  
  
SELECT salario, POW(salario,2) from usuarios;  
  
/* EXP(x) - Recebe um valor como argumento,  
retornando o exponencial do mesmo, */  
  
Select exp(id_usuario) from usuarios;  
  
/* SQRT(x) - Retorna a raiz quadrada de X */  
  
select sqrt(id_usuario) from usuarios;
```

BANCOS DE DADOS

❖ 3.1 Operações de consultas de dados (DQL):

Usando o SQL com operações matemáticas:

```
/* X DIV Y - Resto da divisão de x por y */  
select salario div id_usuario from usuarios;  
  
/* COS(x) - Retorna o cosseno de X */  
Select COS(salario) from usuarios;  
  
/* SIN(X) - Retorna o seno de X */  
Select SIN(salario) from usuarios;  
  
/* TAN(X) - Retorna a tangente de X */  
select tan(salario) from usuarios;
```


BANCOS DE DADOS

❖ 3.1 Operações de consultas de dados (DQL):

Usando o SQL com operadores de concatenação:

```
/* Com o concat podemos unir informações de diferentes colunas  
-com diferentes tipos de dados, varchar, double, date dentro outros */  
  
select concat(<Nome_Do_campo>,'<Símbolo de concatenação>',  
-<Nome_Do_campo>) from <nome_da_tabela>;  
  
select concat(NomeEstado,' - ',SiglaEstado)  
from cad_estado where PKEstado=1;
```

BANCOS DE DADOS

❖ 3.1 Operações de consultas de dados (DQL):

Usando o SQL com operadores de concatenação:

```
/* Com o concat_ws, definimos o separador uma única vez, este é definindo na primeira expressão da função e é assumido para todas as colunas estipuladas */
```

```
select concat_ws('<Símbolo da concatenação>', <Nome_do_campo>,  
<Nome_do_campo>, <Nome_do_campo>) from <nome_da_tabela>;
```

```
select concat_ws(',', PKCliente, NomeCliente, SobrenomeCliente, RG)  
from cad_cliente where PKCliente=1;
```

BANCOS DE DADOS

❖ 3.1 Operações de consultas de dados (DQL):

Usando o SQL com operadores de concatenação:

```
/* Com o group_concat, com ele concatenamos linhas ao invés de colunas,  
transformando o resultado das linhas em uma única coluna */
```

```
select group_concat(<nome_do_campo>) from <nome_da_tabela>  
where <nome_do_campo><operador><valor_da_condição>;
```

```
select group_concat(nomeCliente) from cad_cliente  
where sobreNomeCliente like '%ilva';
```


BANCOS DE DADOS

❖ 3.1 Operações de consultas de dados (DQL):

Usando o SQL com valores nulos:

```
/* Inserir dados em uma tabela com valores nulos */

insert into <Nome_da_Tabela> values
(<valor_do_campo>, '<campo vazio>', <campo_null>);

insert into cad_cliente values (7, ' ', null, null);

/* IS NULL para selecionar uma coluna com valor nulo */

SELECT <Nome_Do_Campo> FROM <nome_da_tabela>
WHERE <nome_da_coluna> IS NULL;

select PKCliente from cad_cliente
where SobreNomeCliente is null;
```

BANCOS DE DADOS

❖ 3.1 Operações de consultas de dados (DQL):

Usando o SQL com valores nulos:

```
/* IS NOT NULL para selecionar uma coluna que NÃO está nula */
```

```
SELECT <Nome_Do_Campo> FROM <nome_da_tabela>  
WHERE <nome_da_coluna> IS NOT NULL;
```

```
select PKCliente from cad_cliente  
where SobreNomeCliente is not null;
```

```
/* COALESCE retorna os valores não nulos, no uso de colunas retorna  
um valor especificado para o campo nulo */
```

```
select coalesce(<nome_da_coluna>,<valor_da_coluna>) from <nome_da_tabela>;
```

```
select coalesce(nomecliente,'Não Cadastrado'), coalesce(rg,0),  
coalesce(sobrenomecliente,'Não cadastrado') from cad_CLIENTE;
```