



## TRABALHO SEMESTRAL BANCO DE DADOS

---

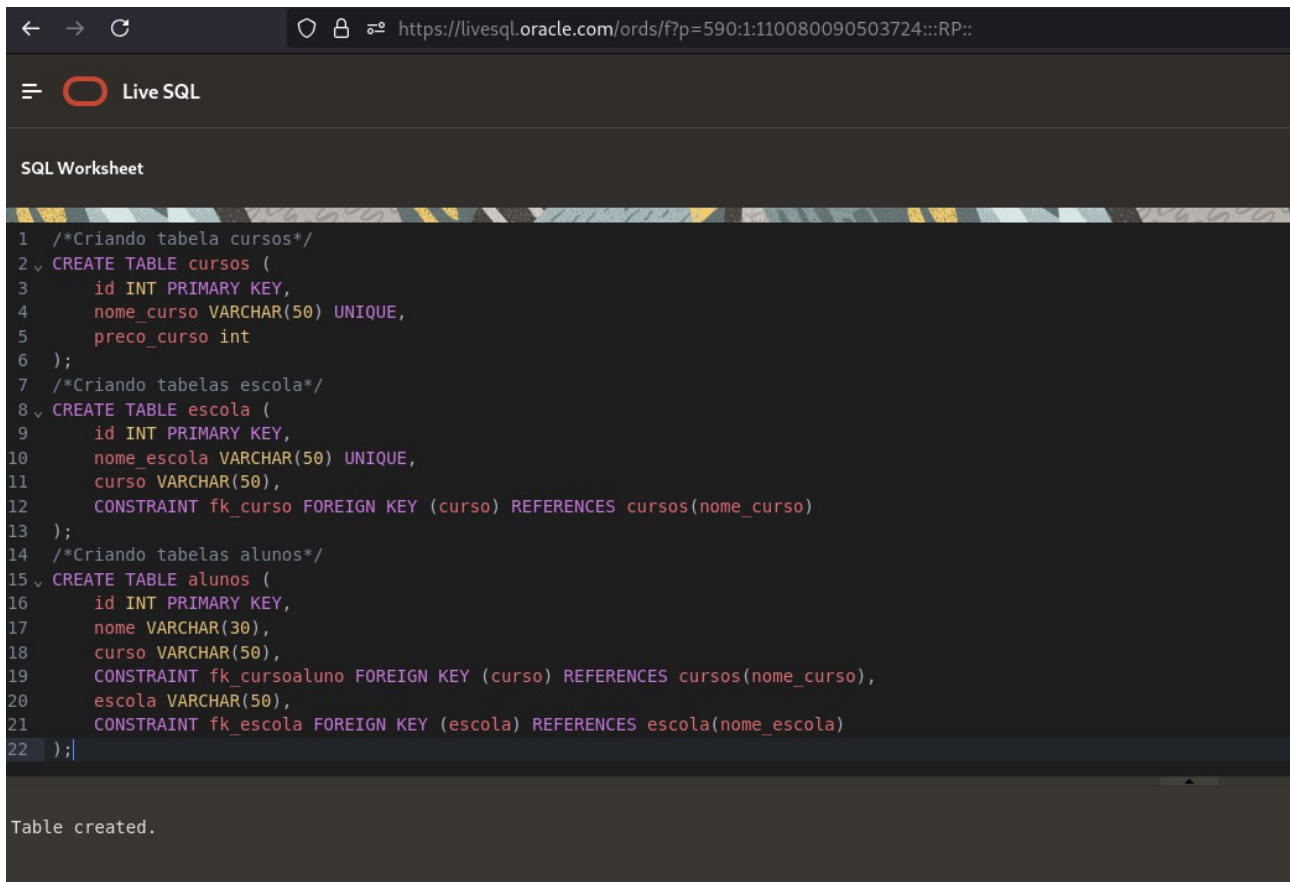
**Nome:** Erick Barbara de Araujo      **RA:** 336356  
**Nome:** Thamara da Silva Sousa      **RA:** 309814  
**Nome:** Vitor Da Silva Bezerra      **RA:** 336459  
**Nome:** Wallace Santos Ribeiro      **RA:** 309767  
**Nome:** Nicolas Roger de Oliveira Aguiar      **RA:** 309431  
**Nome:** Júlia Vecchione Romero      **RA:** 332918

Trabalho semestral deverá ser efetuado em grupo de no mínimo 4 e no máximo 7 pessoas. Este trabalho deverá ser entregue em formato Word(.doc) ou Acrobat (.pdf), deverá conter o código de banco de dados de todos conteúdos que serão ministrados em aula até a data de 22/11/2024, esta atividade deverá ser entregue até 22/11/2024.

A entrega deste trabalho deverá ser efetuada por somente um dos integrantes do grupo, em folha de rosto deverá conter o nome de todos os integrantes do grupo, a forma de correção desta atividade avaliativa se dará de forma que o professor irá copiar o código entregue e executará o mesmo em ferramenta on-line Live Oracle que os alunos estarão habituados, visto que será a mesma utilizada nas aulas.

Nota desta atividade será de no máxima 4, tendo peso equivalente a 40% da nota final.

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **CREATE** e **FOREIGN KEY**.



The screenshot shows a web browser window with the URL <https://livesql.oracle.com/ords/f?p=590:1:110080090503724:::RP::>. The page title is "Live SQL". Below the title, there is a section labeled "SQL Worksheet". The worksheet contains the following SQL code:

```
1 /*Criando tabela cursos*/
2 CREATE TABLE cursos (
3     id INT PRIMARY KEY,
4     nome_curso VARCHAR(50) UNIQUE,
5     preco_curso int
6 );
7 /*Criando tabelas escola*/
8 CREATE TABLE escola (
9     id INT PRIMARY KEY,
10    nome_escola VARCHAR(50) UNIQUE,
11    curso VARCHAR(50),
12    CONSTRAINT fk_curso FOREIGN KEY (curso) REFERENCES cursos(nome_curso)
13 );
14 /*Criando tabelas alunos*/
15 CREATE TABLE alunos (
16     id INT PRIMARY KEY,
17     nome VARCHAR(30),
18     curso VARCHAR(50),
19     CONSTRAINT fk_cursoaluno FOREIGN KEY (curso) REFERENCES cursos(nome_curso),
20     escola VARCHAR(50),
21     CONSTRAINT fk_escola FOREIGN KEY (escola) REFERENCES escola(nome_escola)
22 );
```

Below the code, the message "Table created." is displayed.

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **INSERT INTO**.

```
← → ↺ https://livesql.oracle.com/ords/f?p=590:1:110080090503724::RP::
Live SQL

SQL Worksheet

19     CONSTRAINT fk_cursoaluno FOREIGN KEY (curso) REFERENCES cursos(nome_curso),
20     escola VARCHAR(50),
21     CONSTRAINT fk_escola FOREIGN KEY (escola) REFERENCES escola(nome_escola)
22 );
23
24 /*Inserindo valores e atribuindo dados da chave estrangeira*/
25 INSERT INTO cursos (id, nome_curso, preco_curso) VALUES (1, 'Administração', 440);
26 INSERT INTO escola (id, nome_escola, curso) VALUES (1, 'Escola Barbara de Araujo', 'Administração');
27 INSERT INTO alunos (id, nome, curso, escola) VALUES (1, 'Erick', 'Administração', 'Escola Barbara de Araujo');
28
29 INSERT INTO cursos (id, nome_curso, preco_curso) VALUES (2, 'Teatro', 260);
30 INSERT INTO escola (id, nome_escola, curso) VALUES (2, 'Escola Silva Sousa', 'Teatro');
31 INSERT INTO alunos (id, nome, curso, escola) VALUES (2, 'Thamara', 'Teatro', 'Escola Silva Sousa');
32
33 INSERT INTO cursos (id, nome_curso, preco_curso) VALUES (3, 'Engenharia de Produção', 855);
34 INSERT INTO escola (id, nome_escola, curso) VALUES (3, 'Escola Silva Bezerra', 'Engenharia de Produção');
35 INSERT INTO alunos (id, nome, curso, escola) VALUES (3, 'Victor', 'Engenharia de Produção', 'Escola Silva Bezerra');
36
37 INSERT INTO cursos (id, nome_curso, preco_curso) VALUES (4, 'Paleontologia', 575);
38 INSERT INTO escola (id, nome_escola, curso) VALUES (4, 'Escola Santos Ribeiro', 'Paleontologia');
39 INSERT INTO alunos (id, nome, curso, escola) VALUES (4, 'Wallace', 'Paleontologia', 'Escola Santos Ribeiro');
40
41 INSERT INTO cursos (id, nome_curso, preco_curso) VALUES (5, 'Gestão da Tecnologia da Informação', 710);
42 INSERT INTO escola (id, nome_escola, curso) VALUES (5, 'Escola Roger de Oliveira', 'Gestão da Tecnologia da Informação');
43 INSERT INTO alunos (id, nome, curso, escola) VALUES (5, 'Nicolas', 'Gestão da Tecnologia da Informação', 'Escola Roger de Oliveira');
44
45 INSERT INTO cursos (id, nome_curso, preco_curso) VALUES (6, 'Biologia', 915);
46 INSERT INTO escola (id, nome_escola, curso) VALUES (6, 'Escola Vecchione Romero', 'Biologia');
47 INSERT INTO alunos (id, nome, curso, escola) VALUES (6, 'Julia', 'Biologia', 'Escola Vecchione Romero');

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```


- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **SELECT**.

```
48
49  /*Exibindo dados da tabela*/
50  select * from cursos;
51  select * from escola;
52  select * from alunos;
53
```

ID	NOME_CURSO	PRECO_CURSO
1	Administração	440
2	Teatro	260
3	Engenharia de Produção	855
4	Paleontologia	575
5	Gestão da Tecnologia da Informação	710
6	Biologia	915

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **ROUND**.

← → ↻ <https://livesql.oracle.com/ords/f?p=590:1:110080090503724:::RP::>

≡  Live SQL

SQL Worksheet

```
-- Exibindo dados da tabela cursos com ROUND
SELECT
  id,
  nome_curso AS "Curso",
  ROUND(preco_curso, 2) AS "Preço Arredondado"
FROM cursos;
```


ID	Curso	Preço Arredondado
1	Administração	440
2	Teatro	260
3	Engenharia de Produção	855
4	Paleontologia	575
5	Gestão da Tecnologia da Informação	710
6	Biologia	915

Download CSV

6 rows selected.

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: [AS](#).

← → ↻ <https://livesql.oracle.com/ords/f?p=590:1:110080090503724:::RP::>

≡  Live SQL

SQL Worksheet

```
9 FROM cursos;
10
11 -- Exibindo dados da tabela escola
12 SELECT
13     id AS "ID da Escola",
14     nome_escola AS "Nome da Escola",
15     curso AS "Curso Oferecido"
16 FROM escola;
17
18 -- Exibindo dados da tabela alunos com JOIN e funções ALIAS e ROUND
```

ID da Escola	Nome da Escola	Curso Oferecido
1	Escola Barbara de Araujo	Administração
2	Escola Silva Sousa	Teatro
3	Escola Silva Bezerra	Engenharia de Produção
4	Escola Santos Ribeiro	Paleontologia
5	Escola Roger de Oliveira	Gestão da Tecnologia da Informação
6	Escola Vecchione Romero	Biologia

Download CSV

6 rows selected.

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **COUNT** e **GROUP BY**.

```
67
68 -- Calculando o número de alunos por curso, agrupando os dados por curso
69 SELECT
70     curso AS "Curso",
71     COUNT(id) AS "Número de Alunos"
72 FROM alunos
73 GROUP BY curso;
74
```

Curso	Número de Alunos
Paleontologia	1
Biologia	1
Administração	1
Engenharia de Produção	1
Gestão da Tecnologia da Informação	1
Teatro	1

Download CSV

6 rows selected.

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **HAVING COUNT**.

```
74
75 -- Filtrando os cursos que possuem mais de um aluno matriculado
76 ▾ SELECT
77     curso AS "Curso",
78     COUNT(id) AS "Número de Alunos"
79 FROM alunos
80 GROUP BY curso
81 HAVING COUNT(id) > 1;
```

no data found



- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **AVG, ORDER BY AVG** e **DESC**.

```
83 -- Calculando o preço médio dos cursos e ordenando os resultados por preço médio em ordem decrescente
84 SELECT
85     c.nome_curso AS "Curso",
86     AVG(c.preco_curso) AS "Preço Médio"
87 FROM cursos c
88 GROUP BY c.nome_curso
89 ORDER BY AVG(c.preco_curso) DESC;
```

Curso	Preço Médio
Biologia	915
Engenharia de Produção	855
Gestão da Tecnologia da Informação	710
Paleontologia	575
Administração	440
Teatro	260

Download CSV

6 rows selected.

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **WHERE**, **AND**, **AS** e **ROUND**.

```
90
91 -- Exibindo cursos com preço maior que 500 AND menos de 900
92 ✓ SELECT
93     id,
94     nome_curso AS "Curso",
95     ROUND(preco_curso, 2) AS "Preço Arredondado"
96 FROM cursos
97 WHERE preco_curso > 500 AND preco_curso < 900;
```

ID	Curso	Preço Arredondado
3	Engenharia de Produção	855
4	Paleontologia	575
5	Gestão da Tecnologia da Informação	710

Download CSV

3 rows selected.

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **SELECT, AS** e **JOIN**.

```
98
99 -- Exibindo alunos que estão matriculados em cursos de "Biologia" OR "Teatro"
100 ✓ SELECT
101     a.id AS "ID do Aluno",
102     a.nome AS "Nome do Aluno",
103     a.curso AS "Curso do Aluno",
104     e.nome_escola AS "Escola do Aluno"
105 FROM alunos a
106 JOIN escola e ON a.escola = e.nome_escola
107 WHERE a.curso = 'Biologia' OR a.curso = 'Teatro';
```

ID do Aluno	Nome do Aluno	Curso do Aluno	Escola do Aluno
2	Thamara	Teatro	Escola Silva Sousa
6	Julia	Biologia	Escola Vecchione Romero

Download CSV

2 rows selected.

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **WHERE NOT**.

```
108
109 -- Excluindo cursos com preço superior a 800 usando NOT
110 SELECT
111     id,
112     nome_curso AS "Curso",
113     ROUND(preco_curso, 2) AS "Preço Arredondado"
114 FROM cursos
115 WHERE NOT (preco_curso > 800);
```

ID	Curso	Preço Arredondado
1	Administração	440
2	Teatro	260
4	Paleontologia	575
5	Gestão da Tecnologia da Informação	710

[Download CSV](#)

4 rows selected.

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **JOIN, ON** e **OR**.

```
117 -- Filtrando escolas que oferecem cursos com preço menor que 500 OR mais de 800
118 SELECT
119     e.id AS "ID da Escola",
120     e.nome_escola AS "Nome da Escola",
121     e.curso AS "Curso Oferecido"
122 FROM escola e
123 JOIN cursos c ON e.curso = c.nome_curso
124 WHERE c.preco_curso < 500 OR c.preco_curso > 800;
```

ID da Escola	Nome da Escola	Curso Oferecido
1	Escola Barbara de Araujo	Administração
2	Escola Silva Sousa	Teatro
3	Escola Silva Bezerra	Engenharia de Produção
6	Escola Vecchione Romero	Biologia

Download CSV

4 rows selected.

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **AS** e **WHERE NOT**.

```
126 -- Exibindo alunos que não pertencem à escola "Escola Santos Ribeiro"
```

```
127 SELECT
128     a.id AS "ID do Aluno",
129     a.nome AS "Nome do Aluno",
130     a.curso AS "Curso do Aluno",
131     a.escola AS "Escola"
132 FROM alunos a
133 WHERE NOT (a.escola = 'Escola Santos Ribeiro');
```

ID do Aluno	Nome do Aluno	Curso do Aluno	Escola
1	Erick	Administração	Escola Barbara de Araujo
2	Thamara	Teatro	Escola Silva Sousa
3	Victor	Engenharia de Produção	Escola Silva Bezerra
5	Nicolas	Gestão da Tecnologia da Informação	Escola Roger de Oliveira
6	Julia	Biologia	Escola Vecchione Romero

Download CSV

5 rows selected.

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **LOWER** e **UPPER**.

```
135 -- Utilizando Lower e Upper para visualizanos nome dos alunos em diferentes tamanhos
136 SELECT LOWER(nome) AS nome_minusculo FROM alunos;
137 SELECT UPPER(nome) AS nome_maiusculo FROM alunos;
```

NOME_MINUSCULO
----------------

erick
-------

thamara
---------

victor
--------

wallace
---------

nicolas
---------

julia
-------

Download CSV

6 rows selected.

NOME_MAIUSCULO
----------------

ERICK
-------

THAMARA
---------

VICTOR
--------

WALLACE
---------

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **SELECT**, **AS**, **MAX** e **MIN**.

```
138  
139 -- Maior e menor preço dos cursos  
140 SELECT  
141     MAX(preco_curso) AS "Maior Preço",  
142     MIN(preco_curso) AS "Menor Preço"  
143 FROM cursos;
```

Maior Preço	Menor Preço
915	260

Download CSV





- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **COUNT**, **GROUP BY**, **AVG** e **JOIN**.

```
153 SELECT
154     curso AS "Curso",
155     COUNT(*) AS "Número de Alunos"
156 FROM alunos
157 GROUP BY curso;
158
159 -- Preço médio dos cursos por escola
160 SELECT
161     e.nome_escola AS "Escola",
162     AVG(c.preco_curso) AS "Preço Médio"
163 FROM escola e
164 JOIN cursos c ON e.curso = c.nome_curso
165 GROUP BY e.nome_escola;
```

Curso	Número de Alunos
Paleontologia	1
Biologia	1
Administração	1
Engenharia de Produção	1
Gestão da Tecnologia da Informação	1
Teatro	1

Download CSV

6 rows selected.

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **SELECT, AS, AVG, JOIN, ON** e **GROUP BY**.

```
159 -- Preço médio dos cursos por escola
160 SELECT
161     e.nome_escola AS "Escola",
162     AVG(c.preco_curso) AS "Preço Médio"
163 FROM escola e
164 JOIN cursos c ON e.curso = c.nome_curso
165 GROUP BY e.nome_escola;
```

Escola	Preço Médio
Escola Silva Sousa	260
Escola Barbara de Araujo	440
Escola Roger de Oliveira	710
Escola Vecchione Romero	915
Escola Silva Bezerra	855
Escola Santos Ribeiro	575

Download CSV

6 rows selected.

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **AS**, **WHERE** e **MOD**.

```
166
167 SELECT
168     id AS "ID do Aluno",
169     nome AS "Nome do Aluno",
170     curso AS "Curso do Aluno",
171     escola AS "Escola"
172 FROM alunos
173 WHERE MOD(id, 2) = 0;
```

ID do Aluno	Nome do Aluno	Curso do Aluno	Escola
2	Thamara	Teatro	Escola Silva Sousa
4	Wallace	Paleontologia	Escola Santos Ribeiro
6	Julia	Biologia	Escola Vecchione Romero

Download CSV

3 rows selected.

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **TRUNC** e **MOD**.

```
175 -- Filtrando cursos cujo ID seja divisível por 3 usando MOD
176 SELECT
177     id AS "ID do Curso",
178     nome_curso AS "Curso",
179     TRUNC(preco_curso, 0) AS "Preço Truncado",
180     MOD(id, 3) AS "Resto da Divisão por 3"
181 FROM cursos;
```

ID do Curso	Curso	Preço Truncado	Resto da Divisão por 3
1	Administração	440	1
2	Teatro	260	2
3	Engenharia de Produção	855	0
4	Paleontologia	575	1
5	Gestão da Tecnologia da Informação	710	2
6	Biologia	915	0

Download CSV

6 rows selected.

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **JOIN** e **MOD**.

```
183 -- Calculando preço truncado e exibindo alunos que pertencem a cursos com IDs divisíveis por 2
184 SELECT
185     a.id AS "ID do Aluno",
186     a.nome AS "Nome do Aluno",
187     a.curso AS "Curso do Aluno",
188     e.nome_escola AS "Escola do Aluno",
189     TRUNC(c.preco_curso, 0) AS "Preço Truncado do Curso"
190 FROM alunos a
191 JOIN escola e ON a.escola = e.nome_escola
192 JOIN cursos c ON a.curso = c.nome_curso
193 WHERE MOD(c.id, 2) = 0;
```

ID do Aluno	Nome do Aluno	Curso do Aluno	Escola do Aluno	Preço Truncado do Curso
2	Thamara	Teatro	Escola Silva Sousa	260
4	Wallace	Paleontologia	Escola Santos Ribeiro	575
6	Julia	Biologia	Escola Vecchione Romero	915

Download CSV

3 rows selected.

- Segundo o conteúdo ministrado em aula, nesta parte do código foi utilizado: **ROUND**, **TRUNC** e **MOD**.

```
195 -- Exibindo o preço truncado e arredondado dos cursos e verificando divisibilidade do ID por 5
196 SELECT
197     id AS "ID do Curso",
198     nome_curso AS "Curso",
199     ROUND(preco_curso, 2) AS "Preço Arredondado",
200     TRUNC(preco_curso, 0) AS "Preço Truncado",
201     MOD(id, 5) AS "Resto da Divisão por 5"
202 FROM cursos;
```

ID do Curso	Curso	Preço Arredondado	Preço Truncado	Resto da Divisão por 5
1	Administração	440	440	1
2	Teatro	260	260	2
3	Engenharia de Produção	855	855	3
4	Paleontologia	575	575	4
5	Gestão da Tecnologia da Informação	710	710	0
6	Biologia	915	915	1

Download CSV

6 rows selected.

## Código do Banco de Dados:

```
/*Criando tabela cursos*/
CREATE TABLE cursos (
  id INT PRIMARY KEY,
  nome_curso VARCHAR(50) UNIQUE,
  preco_curso int
);
/*Criando tabelas escola*/
CREATE TABLE escola (
  id INT PRIMARY KEY,
  nome_escola VARCHAR(50) UNIQUE,
  curso VARCHAR(50),
  CONSTRAINT fk_curso FOREIGN KEY (curso) REFERENCES cursos(nome_curso)
);
/*Criando tabelas alunos*/
CREATE TABLE alunos (
  id INT PRIMARY KEY,
  nome VARCHAR(30),
  curso VARCHAR(50),
  CONSTRAINT fk_cursoaluno FOREIGN KEY (curso) REFERENCES cursos(nome_curso),
  escola VARCHAR(50),
  CONSTRAINT fk_escola FOREIGN KEY (escola) REFERENCES escola(nome_escola)
);

/*Inserindo valores e atribuindo dados da chave estrangeira*/
INSERT INTO cursos (id, nome_curso, preco_curso) VALUES (1, 'Administração', 440);
INSERT INTO escola (id, nome_escola, curso) VALUES (1, 'Escola Barbara de Araujo', 'Administração');
INSERT INTO alunos (id, nome, curso, escola) VALUES (1, 'Erick', 'Administração', 'Escola Barbara de Araujo');

INSERT INTO cursos (id, nome_curso, preco_curso) VALUES (2, 'Teatro', 260);
INSERT INTO escola (id, nome_escola, curso) VALUES (2, 'Escola Silva Sousa', 'Teatro');
INSERT INTO alunos (id, nome, curso, escola) VALUES (2, 'Thamara', 'Teatro', 'Escola Silva Sousa');

INSERT INTO cursos (id, nome_curso, preco_curso) VALUES (3, 'Engenharia de Produção', 855);
INSERT INTO escola (id, nome_escola, curso) VALUES (3, 'Escola Silva Bezerra', 'Engenharia de Produção');
INSERT INTO alunos (id, nome, curso, escola) VALUES (3, 'Victor', 'Engenharia de Produção', 'Escola Silva Bezerra');

INSERT INTO cursos (id, nome_curso, preco_curso) VALUES (4, 'Paleontologia', 575);
INSERT INTO escola (id, nome_escola, curso) VALUES (4, 'Escola Santos Ribeiro', 'Paleontologia');
INSERT INTO alunos (id, nome, curso, escola) VALUES (4, 'Wallace', 'Paleontologia', 'Escola Santos Ribeiro');

INSERT INTO cursos (id, nome_curso, preco_curso) VALUES (5, 'Gestão da Tecnologia da Informação', 710);
INSERT INTO escola (id, nome_escola, curso) VALUES (5, 'Escola Roger de Oliveira', 'Gestão da Tecnologia da Informação');
INSERT INTO alunos (id, nome, curso, escola) VALUES (5, 'Nicolas', 'Gestão da Tecnologia da Informação', 'Escola Roger de Oliveira');

INSERT INTO cursos (id, nome_curso, preco_curso) VALUES (6, 'Biologia', 915);
INSERT INTO escola (id, nome_escola, curso) VALUES (6, 'Escola Vecchione Romero', 'Biologia');
INSERT INTO alunos (id, nome, curso, escola) VALUES (6, 'Julia', 'Biologia', 'Escola Vecchione Romero');

/*Exibindo dados da tabela*/
select * from cursos;
select * from escola;
select * from alunos;

-- Exibindo dados da tabela cursos com ROUND
SELECT
  id,
  nome_curso AS "Curso",
  ROUND(preco_curso, 2) AS "Preço Arredondado"
FROM cursos;

-- Exibindo dados da tabela escola
SELECT
  id AS "ID da Escola",
  nome_escola AS "Nome da Escola",
  curso AS "Curso Oferecido"
FROM escola;

-- Calculando o número de alunos por curso, agrupando os dados por curso
SELECT
  curso AS "Curso",
  COUNT(id) AS "Número de Alunos"
FROM alunos
GROUP BY curso;

-- Filtrando os cursos que possuem mais de um aluno matriculado
SELECT
  curso AS "Curso",
  COUNT(id) AS "Número de Alunos"
FROM alunos
GROUP BY curso
HAVING COUNT(id) > 1;

-- Calculando o preço médio dos cursos e ordenando os resultados por preço médio em ordem decrescente
SELECT
  c.nome_curso AS "Curso",
  AVG(c.preco_curso) AS "Preço Médio"
FROM cursos c
GROUP BY c.nome_curso
ORDER BY AVG(c.preco_curso) DESC;

-- Exibindo cursos com preço maior que 500 AND menos de 900
SELECT
  id,
  nome_curso AS "Curso",
  ROUND(preco_curso, 2) AS "Preço Arredondado"
FROM cursos
WHERE preco_curso > 500 AND preco_curso < 900;

-- Exibindo alunos que estão matriculados em cursos de "Biologia" OR "Teatro"
SELECT
  a.id AS "ID do Aluno",
```



```

    a.nome AS "Nome do Aluno",
    a.curso AS "Curso do Aluno",
    e.nome_escola AS "Escola do Aluno"
FROM alunos a
JOIN escola e ON a.escola = e.nome_escola
WHERE a.curso = 'Biologia' OR a.curso = 'Teatro';

-- Excluindo cursos com preço superior a 800 usando NOT
SELECT
    id,
    nome_curso AS "Curso",
    ROUND(preco_curso, 2) AS "Preço Arredondado"
FROM cursos
WHERE NOT (preco_curso > 800);

-- Filtrando escolas que oferecem cursos com preço menor que 500 OR mais de 800
SELECT
    e.id AS "ID da Escola",
    e.nome_escola AS "Nome da Escola",
    e.curso AS "Curso Oferecido"
FROM escola e
JOIN cursos c ON e.curso = c.nome_curso
WHERE c.preco_curso < 500 OR c.preco_curso > 800;

-- Exibindo alunos que não pertencem à escola "Escola Santos Ribeiro"
SELECT
    a.id AS "ID do Aluno",
    a.nome AS "Nome do Aluno",
    a.curso AS "Curso do Aluno",
    a.escola AS "Escola"
FROM alunos a
WHERE NOT (a.escola = 'Escola Santos Ribeiro');

-- Utilizando Lower e Upper para visualizarmos nome dos alunos em diferentes tamanhos
SELECT LOWER(nome) AS nome_minusculo FROM alunos;
SELECT UPPER(nome) AS nome_maiusculo FROM alunos;

-- Maior e menor preço dos cursos
SELECT
    MAX(preco_curso) AS "Maior Preço",
    MIN(preco_curso) AS "Menor Preço"
FROM cursos;

-- Soma total dos preços dos cursos
SELECT
    SUM(preco_curso) AS "Soma Total dos Preços"
FROM cursos;
SELECT
    AVG(preco_curso) AS "Preço Médio"
FROM cursos;

SELECT
    curso AS "Curso",
    COUNT(*) AS "Número de Alunos"
FROM alunos
GROUP BY curso;

-- Preço médio dos cursos por escola
SELECT
    e.nome_escola AS "Escola",
    AVG(c.preco_curso) AS "Preço Médio"
FROM escola e
JOIN cursos c ON e.curso = c.nome_curso
GROUP BY e.nome_escola;

SELECT
    id AS "ID do Aluno",
    nome AS "Nome do Aluno",
    curso AS "Curso do Aluno",
    escola AS "Escola"
FROM alunos
WHERE MOD(id, 2) = 0;

-- Filtrando cursos cujo ID seja divisível por 3 usando MOD
SELECT
    id AS "ID do Curso",
    nome_curso AS "Curso",
    TRUNC(preco_curso, 0) AS "Preço Truncado",
    MOD(id, 3) AS "Resto da Divisão por 3"
FROM cursos;

-- Calculando preço truncado e exibindo alunos que pertencem a cursos com IDs divisíveis por 2
SELECT
    a.id AS "ID do Aluno",
    a.nome AS "Nome do Aluno",
    a.curso AS "Curso do Aluno",
    e.nome_escola AS "Escola do Aluno",
    TRUNC(c.preco_curso, 0) AS "Preço Truncado do Curso"
FROM alunos a
JOIN escola e ON a.escola = e.nome_escola
JOIN cursos c ON a.curso = c.nome_curso
WHERE MOD(c.id, 2) = 0;

-- Exibindo o preço truncado e arredondado dos cursos e verificando divisibilidade do ID por 5
SELECT
    id AS "ID do Curso",
    nome_curso AS "Curso",
    ROUND(preco_curso, 2) AS "Preço Arredondado",
    TRUNC(preco_curso, 0) AS "Preço Truncado",
    MOD(id, 5) AS "Resto da Divisão por 5"
FROM cursos;

```