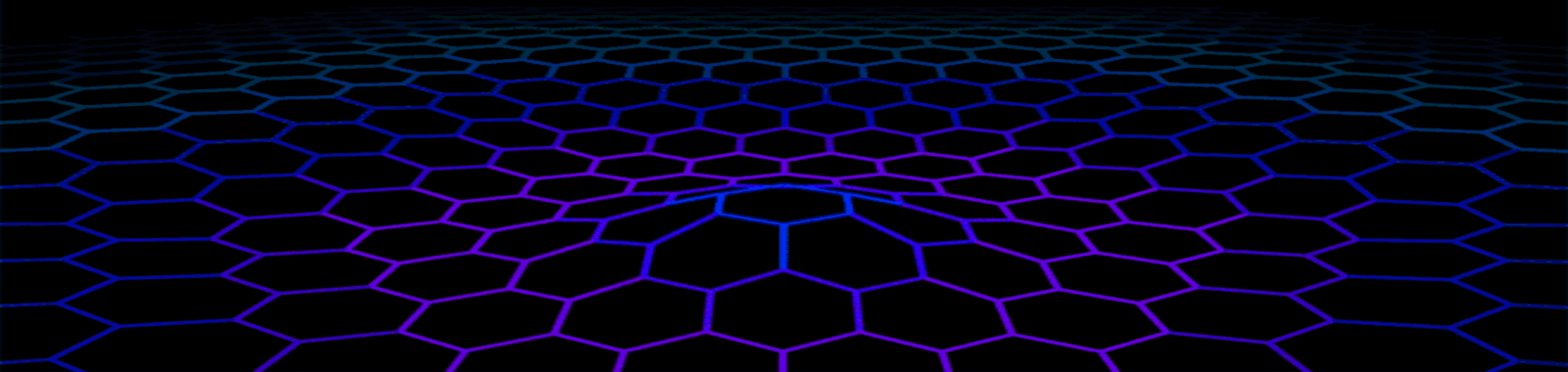
Unisenac
Campus Pelotas

Banco de Dados II



REVISÃO (a saga continua)

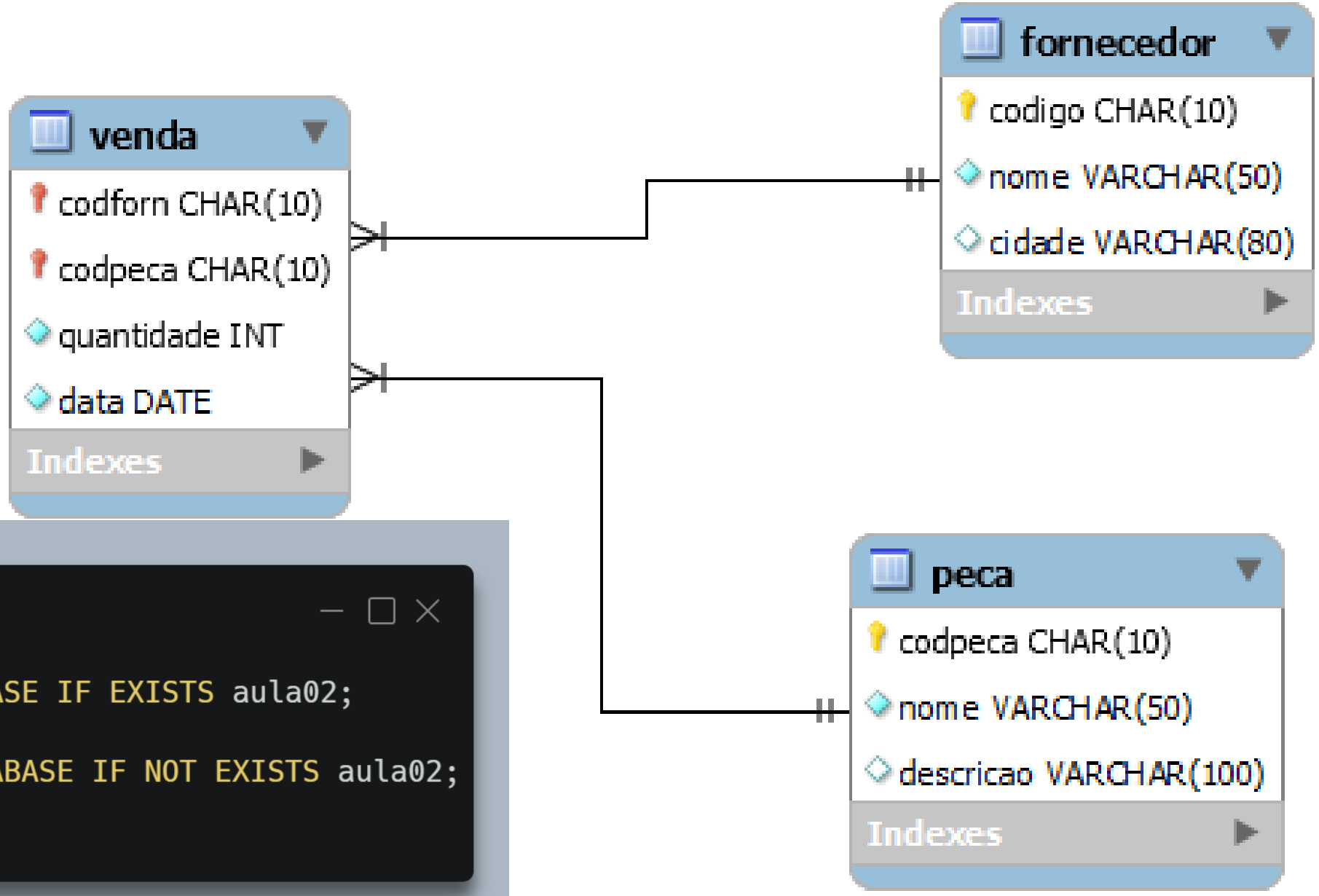
Aquecimento – Exercício 1

Exercício

Defina as tabelas abaixo usando SQL

- **FORNECEDOR** (**codigo**, nome, cidade)
- **VENDA** (**codForn**, **codPeca**, quantidade, dataVenda)
- **PECA** (**codPeca**, nome, descricao)

Aula02

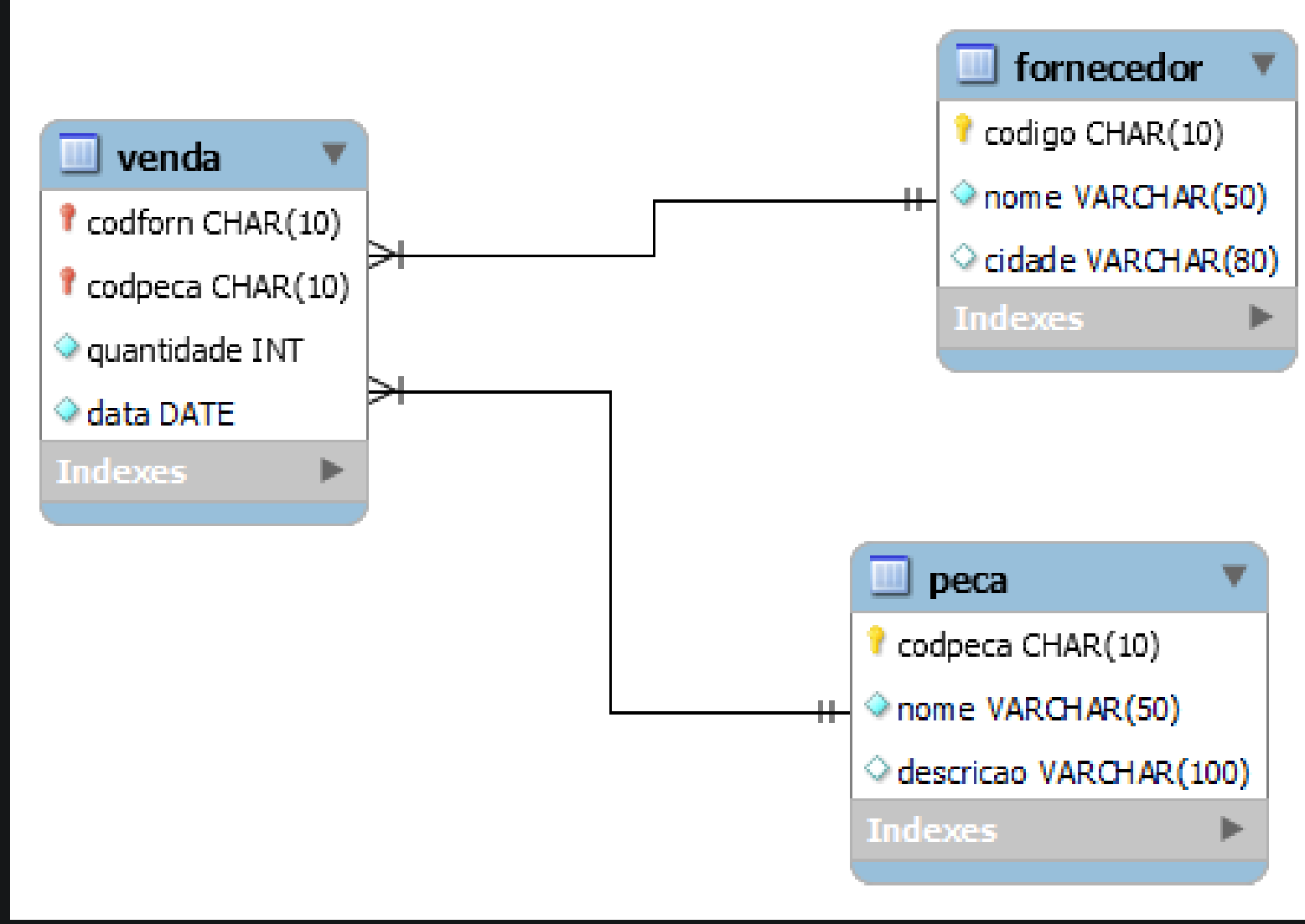



```sql
DROP DATABASE IF EXISTS aula02;

CREATE DATABASE IF NOT EXISTS aula02;

USE aula02;
```

Aula02

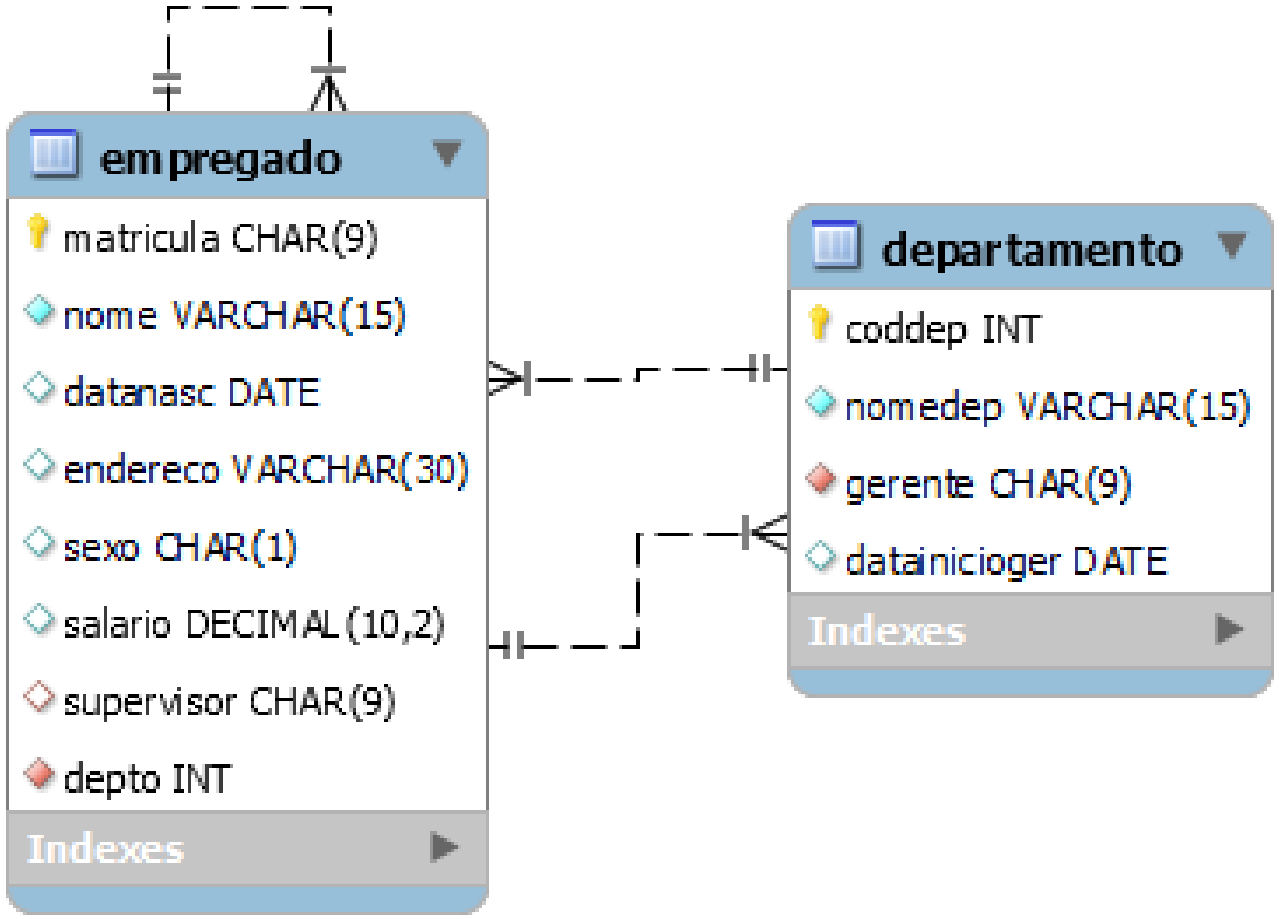
```sql
DROP DATABASE IF EXISTS aula02;

CREATE DATABASE IF NOT EXISTS aula02;

USE aula02;
```

```sql
CREATE TABLE fornecedor
  (
     codigo CHAR(10),
     nome   VARCHAR(50) NOT NULL,
     cidade VARCHAR(80),
     PRIMARY KEY(codigo)
  );

CREATE TABLE peca
  (
     codpeca   CHAR(10),
     nome      VARCHAR(50) NOT NULL,
     descricao VARCHAR(100),
     PRIMARY KEY (codpeca)
  );

CREATE TABLE venda
  (
     codforn    CHAR(10),
     codpeca    CHAR(10),
     quantidade INT NOT NULL,
     data       DATE NOT NULL,
     PRIMARY KEY (codforn, codpeca),
     FOREIGN KEY (codforn) REFERENCES fornecedor(codigo) ON DELETE RESTRICT ON UPDATE CASCADE,
     FOREIGN KEY (codpeca) REFERENCES peca(codpeca) ON DELETE RESTRICT ON UPDATE CASCADE
  );
```



Aquecimento – Exercício 2

Aula02


empregado
matricula CHAR(9)
nome VARCHAR(15)
datanasc DATE
endereco VARCHAR(30)
sexo CHAR(1)
salario DECIMAL(10,2)
supervisor CHAR(9)
depto INT
Indexes
departamento
coddep INT
nomedep VARCHAR(15)
gerente CHAR(9)
datainicioger DATE
Indexes

Empregado

```sql
CREATE TABLE empregado
  (
     matricula  CHAR(9),
     nome       VARCHAR(15) NOT NULL,
     datanasc   DATE,
     endereco   VARCHAR(30),
     sexo       CHAR(1),
     salario    NUMERIC(10, 2),
     supervisor CHAR(9),
     depto      INT(11) NOT NULL
  );
```

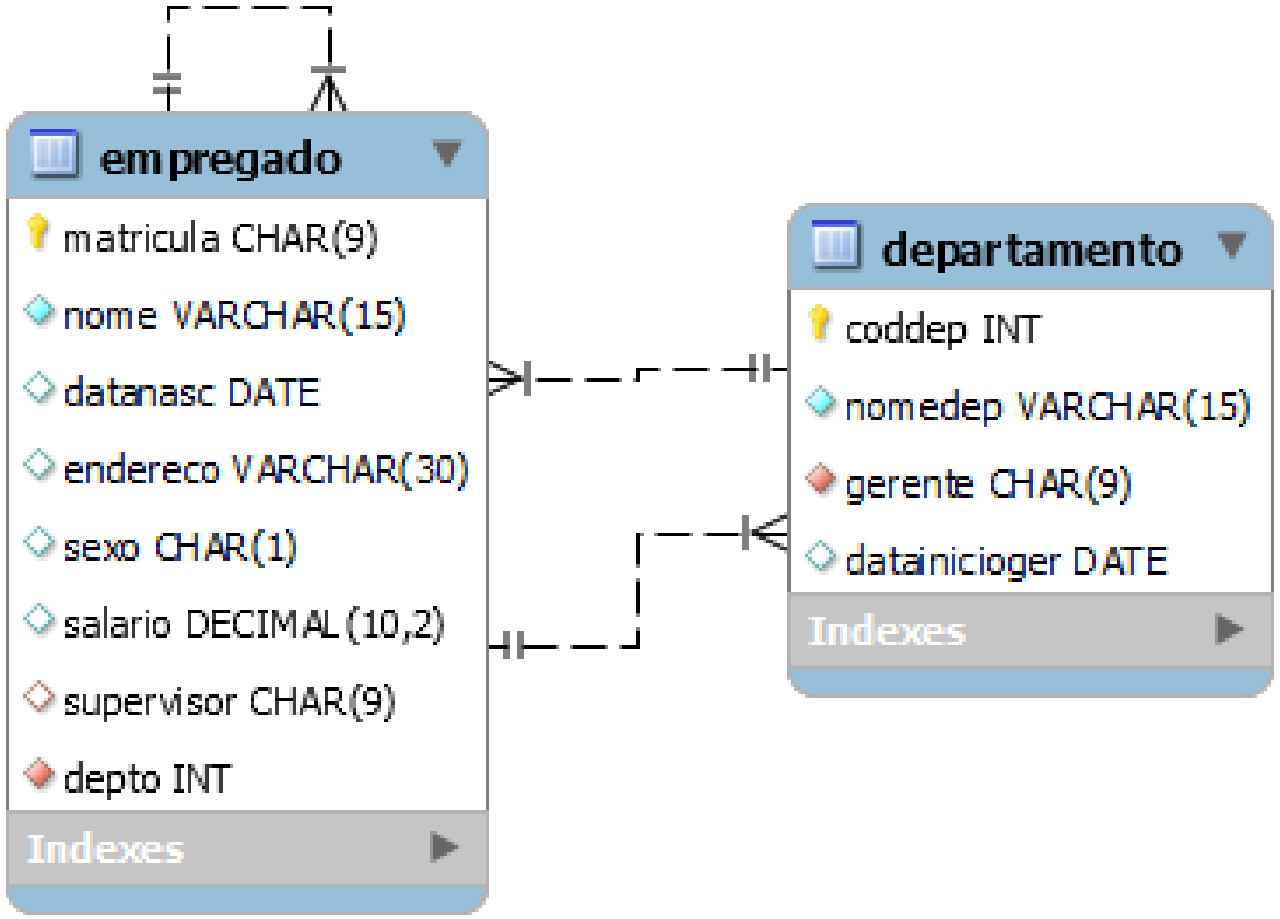
Departamento

```sql
CREATE TABLE departamento
  (
     coddep          INT(11),
     nomedep         VARCHAR(15) NOT NULL,
     gerente         CHAR(9) NOT NULL,
     datainicioger DATE
  );
```

Empregado

```
ALTER TABLE empregado
  ADD CONSTRAINT emppk PRIMARY KEY (matricula);

ALTER TABLE departamento
  ADD CONSTRAINT deppk PRIMARY KEY(coddep);

ALTER TABLE empregado
  ADD CONSTRAINT empsuperfk FOREIGN KEY (supervisor) REFERENCES empregado (
  matricula) ON DELETE RESTRICT ON UPDATE CASCADE;

ALTER TABLE empregado
  ADD CONSTRAINT empdep FOREIGN KEY (depto) REFERENCES departamento(coddep);

ALTER TABLE departamento
  ADD CONSTRAINT depnome UNIQUE (nomedep);

ALTER TABLE departamento
  ADD CONSTRAINT depger FOREIGN KEY (gerente) REFERENCES empregado(matricula);
```

Aula02


empregado
matricula CHAR(9)
nome VARCHAR(15)
datanasc DATE
endereco VARCHAR(30)
sexo CHAR(1)
salario DECIMAL(10,2)
supervisor CHAR(9)
depto INT
Indexes
departamento
coddep INT
nomedep VARCHAR(15)
gerente CHAR(9)
datainicioger DATE
Indexes

Se desejar acompanhar os próximos exemplos...

Se desejar acompanhar os próximos exemplos...

```sql
CREATE TABLE teste
  (
     codigo INT,
     nome   CHAR(15),
     email  VARCHAR(30),
     PRIMARY KEY(codigo)
  );
```

Alteração de tabelas (modificação da estrutura de tabelas)

Quando notamos que as necessidades da aplicação mudaram ou que foi cometido um erro, podemos modificar a estrutura das tabelas já criadas.

Podemos incluir ou excluir colunas, restrições, modificar nome de coluna ou da própria tabela.

Tudo isso pode ser feito através do comando **`ALTER TABLE`**

Alteração de tabelas (modificação da estrutura de tabelas)

ADD <campo> <tipo>

- Insere novo campo

DROP <campo>

- Remove determinado campo

MODIFY <campo><tipo>

- Modifica o tipo de determinado campo

Alteração de tabelas (modificação da estrutura de tabelas)

Inserir na tabela teste o campo endereço após o campo nome.

```
ALTER TABLE teste
     ADD endereco CHAR(50)
     AFTER nome;
```

Observe as alterações com **DESC**RIBE teste;

```
CREATE TABLE teste
  (
     codigo INT,
     nome   CHAR(15),
     email  VARCHAR(30),
     PRIMARY KEY(codigo)
  );
```

Obs. Para inserir antes de todos os outros campos use **FIRST**

```
ALTER TABLE teste
     ADD endereco2 CHAR(50) FIRST;
```

Alteração de tabelas (modificação da estrutura de tabelas)

Inserir na tabela teste o campo nascimento que conterá a data de nascimento dos cadastrados.

```
ALTER TABLE teste ADD nascimento DATE;
```

A nova coluna não pode possuir a restrição de não-nulo, porque a coluna inicialmente deve conter valores nulos. Porém, a restrição de não-nulo pode ser adicionada posteriormente.

Observe as alterações com o `DESCRIBE teste;`

Alteração de tabelas (modificação da estrutura de tabelas)

O campo email foi criado com limite de 30 caracteres. Observe isso com o comando describe teste;

Trocar para 40 caracteres .

```
ALTER TABLE teste MODIFY email CHAR(40);
```

Execute o `DESCRIBE teste;` novamente para observar a alteração.

Alteração de tabelas (modificação da estrutura de tabelas)

Trocar no nome da coluna email por e_mail .

```
ALTER TABLE teste CHANGE email e_mail CHAR(30);
```

Execute o `DESCRIBE teste;` para observar a alteração.

Alteração de tabelas (modificação da estrutura de tabelas)

Abaixo vemos como excluir o campo codigo da tabela Teste:

```
ALTER TABLE teste DROP codigo;
```

Observe as alterações com o `DESCRIBE teste;`

Alteração de tabelas (modificação da estrutura de tabelas)

Definir o campo **nome** como chave da tabela Teste:

```
ALTER TABLE teste ADD PRIMARY KEY (nome);
```

Observe as alterações com o `DESCRIBE teste;`

Alteração de tabelas (modificação da estrutura de tabelas)

Excluir a chave primária, mas não a coluna

```
ALTER TABLE teste DROP PRIMARY KEY;
```

Observe as alterações com o `DESCRIBE teste;`

Alteração de tabelas (modificação da estrutura de tabelas)

Alterar o nome da tabela teste para teste2

```
ALTER TABLE teste RENAME TO teste2;
```

Observe a alteração com **SHOW TABLES;**

Alteração de tabelas (modificação da estrutura de tabelas)

Excluir a tabela. Todos os dados e definições da tabela são removidos.

```
DROP TABLE teste2;
```

Observe a alteração com **SHOW TABLES;**

Atividade

ATIVIDADE - Crie a modelagem lógica, a física e insira dados:

EMPREGADO (matricula, nome, endereco, salario, matriculaBoss, codigoDepartamento)

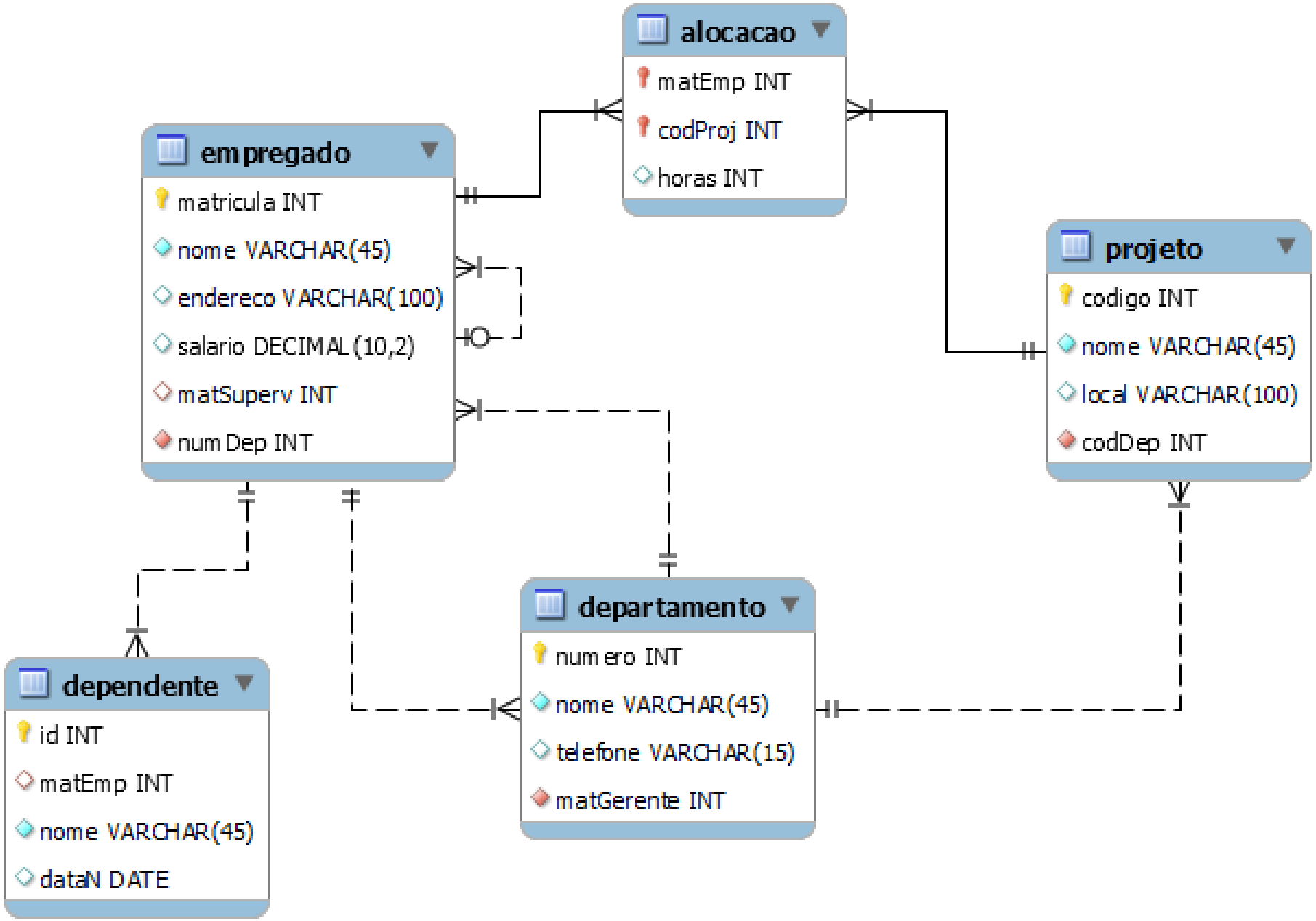
DEPARTAMENTO (codigoDepartamento, nome, matriculaGerente, dataInicio)

PROJETO (codigoProjeto, nome, local, codigoDepartamento)

ALOCACAO (matricula, codigoProjeto, horas)

DEPENDENTE (codigoDependente, matricula, nome, sexo)

Modelo lógico



Modelo físico

```sql
-- Aula02
DROP DATABASE IF EXISTS aula02;
CREATE DATABASE IF NOT EXISTS aula02;
USE aula02;

-- Criar tabela EMPREGADO (matricula, nome, endereco, salario, matriculaBoss, codigoDepartamento)
CREATE TABLE empregado (
  matricula INT,
  nome VARCHAR(50),
  endereco VARCHAR(100),
  salario DECIMAL(10.2),
  matriculaBoss INT,
  codigoDepartamento INT
) ENGINE = INNODB;

-- Criar tabela DEPARTAMENTO (codigoDepartamento, nome, matriculaGerente, dataInicio)
CREATE TABLE departamento (
  codigoDepartamento INT,
  nome VARCHAR(50),
  matriculaGerente INT,
  dataInicio DATE
) ENGINE = INNODB;
```

Modelo físico

```sql
-- Criar tabela DEPARTAMENTO (codigoDepartamento, nome, matriculaGerente, dataInicio)
CREATE TABLE departamento (
  codigoDepartamento INT,
  nome VARCHAR(50),
  matriculaGerente INT,
  dataInicio DATE
) ENGINE = INNODB;

-- Criar tabela PROJETO (codigoProjeto, nome, local, codigoDepartamento)
CREATE TABLE projeto (
  codigoProjeto INT,
  nome VARCHAR(50),
  local VARCHAR(50),
  codigoDepartamento INT
) ENGINE = INNODB;

-- Criar tabela ALOCACAO (matricula, codigoProjeto, horas)
CREATE TABLE alocacao (
  matricula INT,
  codigoProjeto INT,
  horas INT
) ENGINE = INNODB;

-- Criar tabela DEPENDENTE (codigoDependente, matricula, nome, sexo)
CREATE TABLE dependente (
  codigoDependente INT,
  matricula INT,
  nome VARCHAR(50),
  sexo CHAR(1)
) ENGINE = INNODB;
```

Modelo físico

```
47  -- Adicionar chaves primárias
48
49  /* Chave primária da tabela EMPREGADO */
50  ALTER TABLE empregado
51      ADD CONSTRAINT empPK
52      PRIMARY KEY (matricula);
53
54  /* Chave primária da tabela DEPARTAMENTO */
55  ALTER TABLE departamento
56      ADD CONSTRAINT depPK
57      PRIMARY KEY (codigoDepartamento);
58
59  /* Chave primária da tabela PROJETO */
60  ALTER TABLE projeto
61      ADD CONSTRAINT projPK
62      PRIMARY KEY (codigoProjeto);
63
64  /* Chave primária da tabela ALOCACAO
65  (é uma chave composta por dois campos: matricula e codigoProjeto */
66  ALTER TABLE alocacao
67      ADD CONSTRAINT alocPK
68      PRIMARY KEY (matricula, codigoProjeto);
69
70  /* Chave primária da tabela DEPENDENTE */
71  ALTER TABLE DEPENDENTE
72      ADD CONSTRAINT depPK
73      PRIMARY KEY (codigoDependente);
```

Modelo físico

```
75  -- Adicionar constraints da tabela EMPREGADO
76
77  /* Autorelacionamento entre o campo matriculaBoss e matricula */
78  ALTER TABLE empregado
79      ADD CONSTRAINT empBossFK
80      FOREIGN KEY (matriculaBoss) REFERENCES empregado (matricula);
81
82  /* Relacionamento entre as tabelas EMPREGADO e DEPARTAMENTO
83     para indicar a qual departamento o empregado pertence */
84  ALTER TABLE empregado
85      ADD CONSTRAINT empDepFK
86      FOREIGN KEY (codigoDepartamento) REFERENCES departamento (codigoDepartamento);
87
88  -- Adicionar constraints da tabela DEPARTAMENTO
89
90  /* Relacionamento entre as tabelas DEPARTAMENTO e EMPREGADO
91     para indicar qual empregado gerencia o departamento */
92  ALTER TABLE departamento
93      ADD CONSTRAINT depGerFK
94      FOREIGN KEY (matriculaGerente) REFERENCES empregado (matricula);
95
```

Modelo físico

```
96   -- Adicionar constraints da tabela PROJETO
97
98   /* Relacionamento entre as tabelas PROJETO e DEPARTAMENTO
99      para indicar a qual departamento o projeto pertence */
100  ALTER TABLE projeto
101      ADD CONSTRAINT projDepFK
102      FOREIGN KEY (codigoDepartamento) REFERENCES departamento (codigoDepartamento);
103
104  -- Adicionar constraints da tabela ALOCACAO
105
106  /* Relacionamento entre as tabelas ALOCACAO e EMPREGADO
107     para indicar qual empregado está alocado no projeto */
108  ALTER TABLE alocacao
109      ADD CONSTRAINT alocEmpFK
110      FOREIGN KEY (matricula) REFERENCES empregado (matricula);
111
112  /* Relacionamento entre as tabelas ALOCACAO e PROJETO
113     para indicar qual PROJETO faz parte da alocação */
114  ALTER TABLE alocacao
115      ADD CONSTRAINT alocProjFK
116      FOREIGN KEY (codigoProjeto) REFERENCES PROJETO (codigoProjeto);
117
```

Modelo físico

```
118   -- Adicionar constraints da tabela DEPENDENTE
119
120   /* Relacionamento entre as tabelas DEPENDENTE e EMPREGADO
121      para indicar a qual empregado o dependente pertence */
122   ALTER TABLE DEPENDENTE
123       ADD CONSTRAINT depEmpFK
124       FOREIGN KEY (matricula) REFERENCES empregado (matricula);
125
```

Modelo físico

```
mysql> SHOW TABLES;
+------------------+
| Tables_in_aula02 |
+------------------+
| alocacao         |
| departamento     |
| dependente       |
| empregado        |
| projeto          |
+------------------+
5 rows in set (0.00 sec)
```

DML

Crie novamente a tabela **empregado** (pode ser em outra base)

Campo	**Tipo**	**Descrição**
codigo	Int	Código do funcionário(não nulo)
nome	Char(40)	Nome do funcionário (não nulo)
setor	Char(2)	Setor onde o funcionário trabalha
cargo	Char(20)	cargo do funcionário
salario	Decimal(10,2)	salário do funcionário
Chave Primária		campo codigo

Crie a tabela **empregado**

```sql
DROP DATABASE IF EXISTS aula02b;
CREATE DATABASE IF NOT EXISTS aula02b;
USE aula02b

DROP TABLE IF EXISTS empregado;

CREATE TABLE IF NOT EXISTS empregado(
    codigo INT,
    nome CHAR(40) NOT NULL,
    setor CHAR(2),
    cargo CHAR(20),
    salario decimal(10,2),
    PRIMARY KEY(codigo)
);
```

Inserindo registros

codigo	nome	setor	cargo	salário
1	Edecio Muito Legal	2	Designer	1000
3	Eduardo Monks	5	Dev	1500
4	Raquel Murillo	4	Dev	1500
6	Gladimau Catarino	4	Analista	2200
7	Sergio Marquina	4	Boss	9900
9	Alicia Sierra	5	Boss	9900
10	Angelo Light	1	Dev	1500
15	Silene Oliveira	1	DBA	2500
25	Darta Inhame	3	Designer	1650

Para se adicionar dados a uma tabela, usamos o comando INSERT:

```
INSERT INTO empregado VALUES (1,"Edecio Muito Legal","2","Designer",1000);
```

Inserindo registros

```
INSERT INTO empregado VALUES (1,”Edecio Muito Legal”,”2”,”Designer”,1000);
INSERT INTO empregado VALUES (3,”Eduardo Monks”,”5”,”Dev”,1500);
INSERT INTO empregado VALUES (4,”Raquel Murillo”,”4”,”Dev”,1500);
INSERT INTO empregado VALUES (6,”Gladimau Catarino”,”4”,”Analista”,2200);
INSERT INTO empregado VALUES (7,”Sergio Marquina”,”4”,”Boss”,9900);
INSERT INTO empregado VALUES (9,”Alicia Sierra”,”5”,”Boss”,9900);
INSERT INTO empregado VALUES (10,”Angelo Light”,”1”,”Dev”,1500);
INSERT INTO empregado VALUES (15,”Silene Oliveira”,”1”,”DBA”,2500);
INSERT INTO empregado VALUES (25,”Darta Inhame”,”3”,”Designer”,1650);
```

Listar todos os campos(*) de todos os registros da tabela empregado

Listar todos os campos(*) de todos os registros da tabela empregado

```
SELECT * FROM empregado;
```

```
mysql> SELECT * FROM empregado;
+--------+--------------------+-------+----------+---------+
| codigo | nome               | setor | cargo    | salario |
+--------+--------------------+-------+----------+---------+
|      1 | Edecio Muito Legal | 2     | Designer | 1000.00 |
|      3 | Eduardo Monks      | 5     | Dev      | 1500.00 |
|      4 | Raquel Murillo     | 4     | Dev      | 1500.00 |
|      6 | Gladimau Catarino  | 4     | Analista | 2200.00 |
|      7 | Sergio Marquina    | 4     | Boss     | 9900.00 |
|      9 | Alicia Sierra      | 5     | Boss     | 9900.00 |
|     10 | Angelo Light       | 1     | Dev      | 1500.00 |
|     15 | Silene Oliveira    | 1     | DBA      | 2500.00 |
|     25 | Darta Inhame       | 3     | Designer | 1650.00 |
+--------+--------------------+-------+----------+---------+
9 rows in set (0.00 sec)
```

Listar os campos codigo e nome de todos registros da tabela empregado.

Listar os campos codigo e nome de todos registros da tabela empregado.

```
SELECT codigo, nome FROM empregado;
```

```
mysql> SELECT codigo, nome FROM empregado;
+--------+--------------------+
| codigo | nome               |
+--------+--------------------+
|      1 | Edecio Muito Legal |
|      3 | Eduardo Monks      |
|      4 | Raquel Murillo     |
|      6 | Gladimau Catarino  |
|      7 | Sergio Marquina    |
|      9 | Alicia Sierra      |
|     10 | Angelo Light       |
|     15 | Silene Oliveira    |
|     25 | Darta Inhame       |
+--------+--------------------+
9 rows in set (0.00 sec)
```

Listar todos os registros da tabela empregado que possuírem “Sergio Marquina” no campo nome.

Listar todos os registros da tabela empregado que possuírem “Sergio Marquina” no campo nome.

```
SELECT * FROM empregado WHERE (nome = ”Sergio Marquina”);
```

```
mysql> SELECT * FROM empregado WHERE (nome = "Sergio Marquina");
+--------+-----------------+-------+-------+---------+
| codigo | nome            | setor | cargo | salario |
+--------+-----------------+-------+-------+---------+
|      7 | Sergio Marquina | 4     | Boss  | 9900.00 |
+--------+-----------------+-------+-------+---------+
1 row in set (0.00 sec)
```

UPDATE

Procurar na tabela registros que contenham no campo **nome** o conteúdo **“Sergio Marquina”**, e substituir por “**Salvador Martin**”.

UPDATE (SEMPRE com **WHERE**)

Procurar na tabela registros que contenham no campo **nome** o conteúdo **“Sergio Marquina”**, e substituir por “**Salvador Martin**”.

```
UPDATE empregado SET nome = ”Salvador Martin” WHERE nome = ”Sergio Marquina”;
```

```
mysql> UPDATE empregado SET nome = "Salvador Martin" WHERE nome = "Sergio Marquina";
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM empregado;
+--------+--------------------+-------+----------+---------+
| codigo | nome               | setor | cargo    | salario |
+--------+--------------------+-------+----------+---------+
|      1 | Edecio Muito Legal | 2     | Designer | 1000.00 |
|      3 | Eduardo Monks      | 5     | Dev      | 1500.00 |
|      4 | Raquel Murillo     | 4     | Dev      | 1500.00 |
|      6 | Gladimau Catarino  | 4     | Analista | 2200.00 |
|      7 | Salvador Martin    | 4     | Boss     | 9900.00 |
|      9 | Alicia Sierra      | 5     | Boss     | 9900.00 |
|     10 | Angelo Light       | 1     | Dev      | 1500.00 |
|     15 | Silene Oliveira    | 1     | DBA      | 2500.00 |
|     25 | Darta Inhame       | 3     | Designer | 1650.00 |
+--------+--------------------+-------+----------+---------+
9 rows in set (0.00 sec)
```

DELETE

Deletar, da tabela empregado, todos os registros que têm o conteúdo ‘**3**’ no campo setor.

DELETE (SEMPRE com **WHERE**)

Deletar, da tabela empregado, todos os registros que têm o conteúdo '**3**' no campo setor.

```
DELETE FROM empregado WHERE (setor = "3");
```

```
mysql> DELETE FROM empregado WHERE (setor = "3");
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM empregado;
+--------+--------------------+-------+----------+---------+
| codigo | nome               | setor | cargo    | salario |
+--------+--------------------+-------+----------+---------+
|      1 | Edecio Muito Legal | 2     | Designer | 1000.00 |
|      3 | Eduardo Monks      | 5     | Dev      | 1500.00 |
|      4 | Raquel Murillo     | 4     | Dev      | 1500.00 |
|      6 | Gladimau Catarino  | 4     | Analista | 2200.00 |
|      7 | Sergio Marquina    | 4     | Boss     | 9900.00 |
|      9 | Alicia Sierra      | 5     | Boss     | 9900.00 |
|     10 | Angelo Light       | 1     | Dev      | 1500.00 |
|     15 | Silene Oliveira    | 1     | DBA      | 2500.00 |
+--------+--------------------+-------+----------+---------+
8 rows in set (0.00 sec)
```

ORDER BY

Listar os empregados do setor 1, em ordem alfabética de cargo.

ORDER BY

Listar os empregados do setor 1, em ordem alfabética de cargo.

```
SELECT * FROM empregado WHERE (setor = "1") ORDER BY cargo;
```

```
mysql> SELECT * FROM empregado WHERE (setor = "1") ORDER BY cargo;
+--------+-----------------+-------+-------+---------+
| codigo | nome            | setor | cargo | salario |
+--------+-----------------+-------+-------+---------+
|     15 | Silene Oliveira | 1     | DBA   | 2500.00 |
|     10 | Angelo Light    | 1     | Dev   | 1500.00 |
+--------+-----------------+-------+-------+---------+
2 rows in set (0.34 sec)
```

ORDER BY

Obtenha a listagem abaixo:

```
mysql> SELECT * FROM empregado ORDER BY cargo DESC, nome ASC;
+--------+-------------------+-------+----------+---------+
| codigo | nome              | setor | cargo    | salario |
+--------+-------------------+-------+----------+---------+
|     10 | Angelo Light      | 1     | Dev      | 1500.00 |
|      3 | Eduardo Monks     | 5     | Dev      | 1500.00 |
|      4 | Raquel Murillo    | 4     | Dev      | 1500.00 |
|      1 | Edecio Muito Legal| 2     | Designer | 1000.00 |
|     15 | Silene Oliveira   | 1     | DBA      | 2500.00 |
|      9 | Alicia Sierra     | 5     | Boss     | 9900.00 |
|      7 | Sergio Marquina   | 4     | Boss     | 9900.00 |
|      6 | Gladimau Catarino | 4     | Analista | 2200.00 |
+--------+-------------------+-------+----------+---------+
8 rows in set (0.00 sec)
```

ORDER BY

Obtenha a listagem abaixo:

```
SELECT * FROM empregado ORDER BY cargo DESC, nome ASC;
```

```
mysql> SELECT * FROM empregado ORDER BY cargo DESC, nome ASC;
+--------+--------------------+-------+----------+---------+
| codigo | nome               | setor | cargo    | salario |
+--------+--------------------+-------+----------+---------+
|     10 | Angelo Light       | 1     | Dev      | 1500.00 |
|      3 | Eduardo Monks      | 5     | Dev      | 1500.00 |
|      4 | Raquel Murillo     | 4     | Dev      | 1500.00 |
|      1 | Edecio Muito Legal | 2     | Designer | 1000.00 |
|     15 | Silene Oliveira    | 1     | DBA      | 2500.00 |
|      9 | Alicia Sierra      | 5     | Boss     | 9900.00 |
|      7 | Sergio Marquina    | 4     | Boss     | 9900.00 |
|      6 | Gladimau Catarino  | 4     | Analista | 2200.00 |
+--------+--------------------+-------+----------+---------+
8 rows in set (0.00 sec)
```

DML

EXERCÍCIOs

1. Apresentar a listagem completa dos registros da tabela **empregado**;
2. Apresentar uma listagem dos **nomes** e dos **cargos** de todos os registros da tabela **empregado**;
3. Apresentar uma listagem dos **nomes** dos empregados do **setor 1**
4. Listagem dos **nomes** e dos **salários** por **ordem** de **nome** (**A-Z**)
5. Listagem dos **nomes** e dos **salários** por **ordem** de **nome** em formato descendente (**Z-A**)
6. Listagem dos **setores** e **nomes** colocados por **ordem** do campo **setor** em formato **ascendente** e do campo **nome** em formato **descendente**.
7. Listagem de **nomes ordenados** pelo campo **nome** em formato **ascendente**, dos empregados do **setor 4**.
8. O empregado de **código 6** (o melhor de todos) teve um **aumento** de **salário**. Seu salário passa a ser **8000**.
9. **Eduardo Monks** foi **transferido** do departamento 5 para o **departamento 3**.
10. **Todos** os **empregados** da empresa tiveram um **aumento** de salário de **20%**.
11. **Todos** os **empregados** do **setor 3** foram demitidos , **exclua-os**.
12. **Gladimau Catarino** pediu demissão**, exclua-o**.

EXERCÍCIOs

1. Inserir na tabela empregado o campo **admissao** que conterá a data de admissão dos empregados.
2. Em seguida será necessário atualizar a tabela com as datas de admissão dos empregados ativos.

codigo	nome	setor	cargo	salário	admissao
1	Edecio Muito Legal	2	Designer	1000	01/04/2000
4	Raquel Murillo	4	Dev	1500	01/07/2000
7	Sergio Marquina	4	Boss	9900	20/09/2012
9	Alicia Sierra	5	Boss	9900	20/08/2000
10	Angelo Light	1	Dev	1500	01/06/2000
15	Silene Oliveira	1	DBA	2500	01/06/2000

EXERCÍCIOs

1. Apresente a listagem dos empregados que foram **admitidos em 01/06/2000**
2. Apresente a listagem dos funcionários que foram **admitidos após 01/01/2002**
3. O **departamento 3** foi reaberto e admitiu-se os seguintes empregados:

codigo	nome	setor	cargo	salário	admissao
16	Eduardo Monks	3	Boss	9900	19/08/2020
22	Darta Inhame	3	Dev	1500	19/08/2020
29	Hepa Tite	3	Designer	1450	19/08/2020

EXERCÍCIOs

1. Apresentar **nome** e **salário** dos empregados que **ganham acima** de **2500.00**
2. Listar os **empregados** do **setor 2**
3. Listar os **empregados** cujo **cargo** é **Dev**
4. Listar **empregados** com **salário até 2000**
5. Listar **Dev** do **setor 4**
6. Listar empregados que sejam **Boss** ou **DBA**
7. Listar empregados que **não** sejam **Boss**

BETWEEN

```
SELECT *
FROM empregado
WHERE salario BETWEEN 2000 AND 8000;
```

EXERCÍCIOs

1. Listar empregados com salário **entre 1700** e **2000**
2. Listar todos os **Dev** e **Boss**

LIKE

Para verificar sequência de caracteres dentro de um campo do tipo `STRING (CHAR ou VARCHAR)`, pode-se utilizar junto com a clausula `WHERE` uma condição baseada no uso do operador `LIKE`.

<expressão> [NOT] LIKE <valor>

Exemplos:

```
SELECT * FROM empregado WHERE nome LIKE 'E%’; -- Lista empregados que o nome começa com a letra E
SELECT * FROM empregado WHERE nome LIKE '_A%'; -- Lista empregados que a segunda letra do nome é A
SELECT * FROM empregado WHERE nome LIKE '__A%'; -- Lista empregados que a terceira letra do nome é A
SELECT * FROM empregado WHERE nome LIKE '%A’; -- Lista empregados que última letra do nome é A
```

```
mysql> SELECT * FROM empregado WHERE nome LIKE 'E%';
+--------+-------------------+-------+----------+---------+
| codigo | nome              | setor | cargo    | salario |
+--------+-------------------+-------+----------+---------+
|      1 | Edecio Muito Legal | 2     | Designer | 1000.00 |
|      3 | Eduardo Monks     | 5     | Dev      | 1500.00 |
+--------+-------------------+-------+----------+---------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM empregado WHERE nome LIKE '_A%';
+--------+----------------+-------+-------+---------+
| codigo | nome           | setor | cargo | salario |
+--------+----------------+-------+-------+---------+
|      4 | Raquel Murillo | 4     | Dev   | 1500.00 |
|      7 | Salvador Martin | 4     | Boss  | 9900.00 |
+--------+----------------+-------+-------+---------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM empregado WHERE nome LIKE '__A%';
+--------+-----------------+-------+----------+---------+
| codigo | nome            | setor | cargo    | salario |
+--------+-----------------+-------+----------+---------+
|      6 | Gladimau Catarino | 4     | Analista | 2200.00 |
+--------+-----------------+-------+----------+---------+
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM empregado WHERE nome LIKE '%A';
+--------+----------------+-------+-------+---------+
| codigo | nome           | setor | cargo | salario |
+--------+----------------+-------+-------+---------+
|      9 | Alicia Sierra  | 5     | Boss  | 9900.00 |
|     15 | Silene Oliveira | 1     | DBA   | 2500.00 |
+--------+----------------+-------+-------+---------+
2 rows in set (0.00 sec)
```

EXERCÍCIOs

1. Listar empregados cujo **nome comece** com a letra **A**
2. Listar empregados cujo **nome** tem a **segunda** letra **A**
3. Listar empregados que tem a sequência **AN** em **qualquer posição** do **nome**.

Algumas FUNÇÕES

`AVG()` - média aritmética

`MAX()` - Maior valor

`MIN()` - Menor valor

`SUM()` - Soma dos valores

`COUNT()` - Número de valores

- `ALL` - contagem dos valores não vazios
- `DISTINCT` - contagem dos valores não vazios e únicos

Exemplos

1. Exiba o código, o nome e o salário dos empregados que tiverem o menor salário.

```
SELECT codigo, nome, salario
FROM empregado
WHERE salario =(SELECT MIN(salario) FROM empregado);
```

2. Exiba o código, o nome e o salário dos empregados que tiverem o maior salário.

```
SELECT codigo, nome, salario
FROM empregado
WHERE salario =(SELECT MAX(salario) FROM empregado);
```

```
mysql> SELECT codigo, nome, salario
    -> FROM empregado
    -> WHERE salario =(SELECT MIN(salario) FROM empregado);
+--------+--------------------+---------+
| codigo | nome               | salario |
+--------+--------------------+---------+
|      1 | Edecio Muito Legal | 1000.00 |
+--------+--------------------+---------+
1 row in set (0.00 sec)
```

```
mysql> SELECT codigo, nome, salario
    -> FROM empregado
    -> WHERE salario =(SELECT MAX(salario) FROM empregado);
+--------+-----------------+---------+
| codigo | nome            | salario |
+--------+-----------------+---------+
|      7 | Salvador Martin | 9900.00 |
|      9 | Alicia Sierra   | 9900.00 |
+--------+-----------------+---------+
2 rows in set (0.00 sec)
```

EXERCÍCIOs

1. **Média** aritmética dos **salários** de todos os empregados
2. **Média** aritmética dos **salários** de todos os empregados do **setor 2**
3. **Soma** dos **salários** de todos os empregados
4. **Soma** dos **salários** de todos os empregados do **setor 5**
5. **Maior salário** existente entre todos os empregados
6. **Menor salário** existente entre todos os empregados
7. **Numero** de **empregados** do **setor 3**
8. **Número** de **empregados** que ganham **mais que 2000,00**
9. **Número** de **setores** existentes no cadastro de empregados.

Crie a tabela **cliente**

```
CREATE TABLE cliente
(
    codigo    CHAR(3) PRIMARY KEY,
    nome      VARCHAR(40) NOT NULL,
    endereco  VARCHAR(50) NOT NULL,
    cidade    VARCHAR(20) NOT NULL,
    uf        CHAR(2) NOT NULL,
    cep       CHAR(9) NOT NULL
);
```

Crie a tabela **conta**

```
CREATE TABLE conta
(
    numero      CHAR(6) PRIMARY KEY,
    valor       DECIMAL(10,2) NOT NULL,
    vencimento  DATE NOT NULL,
    codcli      CHAR(3) NOT NULL
);
```

conta			
numero	valor	vencimento	codcli
A12345	128447	21/10/2016	221
A12346	228400	22/10/2016	111
A12347	328450	23/09/2015	331
B12348	428447	24/10/2015	331
B22349	528429	10/09/2016	331
B32341	628447	11/10/2015	221
C42340	23322	10/09/2015	331
C52342	28447	11/10/2016	555
Y17133	1471	23/10/2016	555
Z17144	2	24/09/2015	221

cliente					
codigo	nome	endereco	cidade	uf	cep
111	Dedeh	Andrade Neves	Pelotas	RS	90000
221	Angelo	Bento	Pelotas	RS	91000
331	Roberto	Bento	Pelotas	RS	91000
213	Carlinhos	Amarante	Campinas	SP	20000
444	Gladislau	Donja	Pelotas	RS	93000
555	Pablo Escobar	Laranjal	Pelotas	RS	90000
664	Tio da Sukita	Pampas	Herval	RS	70000

Consulta simples (em tabela única)

Imagine a necessidade de obter uma relação das contas existentes e seus respectivos clientes.

```
SELECT numero, codcli
FROM conta;
```

Simples pois tanto o campo **numero** como **codcli** estão na mesma tabela, mas imagine agora que eu quero exibir o nome do cliente e não o código

numero	codcli
A12345	221
A12346	111
A12347	331
B12348	331
B22349	331
B32341	221
C42340	331
C52342	555
Y17133	555
Z17144	221

Consulta em mais de uma tabela

Obter a relação das contas existentes (tabela conta) e o nome dos clientes (tabela cliente) que possuem essas contas:

```
SELECT conta.numero, cliente.nome
FROM cliente, conta
WHERE cliente.codigo = conta.codcli;
```

numero	nome
A12345	Angelo
A12346	Dedeh
A12347	Roberto
B12348	Roberto
B22349	Roberto
B32341	Angelo
C42340	Roberto
C52342	Pablo Escobar
Y17133	Pablo Escobar
Z17144	Angelo

Esse comando pode ser entendido como selecione os campos **nome** de cliente e **numero** da união das tabelas **cliente** e **conta** onde o **codigo** de **cliente** seja igual ao **codcli** de **conta**.

EXERCÍCIOs

1. Apresentação de uma listagem ordenada por nomes de clientes, mostrando a relação de contas que cada um possui e seus respectivos valores.

2. Listagem que apresente as contas existentes do cliente “Roberto”. Na listagem devem constar o nome do cliente, o numero da conta e seu valor correspondente.

3. Apresentar os nomes dos clientes e a data de vencimento de todas as contas do mês de setembro de 2016. A listagem deve ser apresentada na ordem cronológica de vencimento.

4. Apresentação do nome dos clientes e de todas as contas que possuem vencimento no mês de outubro de qualquer ano.

GROUP BY

Obter a quantidade de contas existente de cada cliente.

Para solucionar esta necessidade, deve-se utilizar junto a **WHERE** a cláusula **GROUP BY**

```
SELECT codcli, COUNT(*)
FROM conta
GROUP BY codcli;
```

codcli	COUNT(*)
111	1
221	3
331	4
555	2

GROUP BY

Para exibir o nome do cliente e não o código:

```
SELECT cliente.nome, COUNT(*)
FROM cliente, conta
WHERE cliente.codigo = conta.codcli
GROUP BY cliente.nome;
```

```
+---------------+----------+
| nome          | COUNT(*) |
+---------------+----------+
| Angelo        |        3 |
| Dedeh         |        1 |
| Pablo Escobar |        2 |
| Roberto       |        4 |
+---------------+----------+
```

GROUP BY

Podemos usar a cláusula GROUP BY para calcular a média de contas para cada elemento de um cliente.

Exemplo:

```
SELECT codcli, AVG(valor)
FROM conta
GROUP BY codcli;
```

codcli	AVG(valor)
111	2284.000000
221	2522.986667
331	3271.620000
555	149.590000

Observe que quando usamos GROUP BY a função AVG() retorna a média calculada para cada componente do grupo especificado, no caso "codcli".

GROUP BY

Podemos também testar o valor retornado por avg():
Por exemplo: Exibir os clientes que possuem a média dos valores das contas maior que 2000

```
SELECT codcli, AVG(valor)
FROM conta
GROUP BY codcli
HAVING AVG(valor)>2000;
```

```
+--------+-------------+
| codcli | AVG(valor)  |
+--------+-------------+
| 111    | 2284.000000 |
| 221    | 2522.986667 |
| 331    | 3271.620000 |
+--------+-------------+
```

Note que a cláusula having usada aqui tem o mesmo significado que where nas consultas normais.

OBS: a cláusula “WHERE” não pode ser usada para restringir **grupos** que deverão ser exibidos. Acompanhando o GROUP BY utilizamos a cláusula “**HAVING**”.

ROLLUP

Quando se utiliza a cláusula GROUP BY, como por exemplo, totalizar as contas por cliente, o gerenciador retorna a consulta com uma linha para cada cliente. O modificador **ROLLUP** faz com que o MySQL retorne também as linhas totalizadas, ou seja, o total por cliente e o total geral.

Exemplo:

```
SELECT codcli, COUNT(valor), SUM(valor), AVG(valor)
FROM conta
GROUP BY codcli
WITH ROLLUP;
```

codcli	COUNT(valor)	SUM(valor)	AVG(valor)
111	1	2284.00	2284.000000
221	3	7568.96	2522.986667
331	4	13086.48	3271.620000
555	2	299.18	149.590000
NULL	10	23238.62	2323.862000

ROLLUP

```
+--------+------------+
| codcli | SUM(valor) |
+--------+------------+
| 111    |    2284.00 |
| 221    |    7568.96 |
| 331    |   13086.48 |
| 555    |     299.18 |
+--------+------------+
4 rows in set (0.01 sec)
```

```
SELECT codcli, SUM(valor)
FROM conta
GROUP BY codcli;
```

```
+--------+------------+
| codcli | SUM(valor) |
+--------+------------+
| 111    |    2284.00 |
| 221    |    7568.96 |
| 331    |   13086.48 |
| 555    |     299.18 |
| NULL   |   23238.62 |
+--------+------------+
5 rows in set (0.00 sec)
```

```
SELECT codcli,SUM(valor)
FROM conta
GROUP BY codcli
WITH ROLLUP;
```

ROLLUP

Angelo	A12345	1284.47
Angelo	B32341	6284.47
Angelo	Z17144	0.02
Angelo	NULL	7568.96
Dedeh	A12346	2284.00
Dedeh	NULL	2284.00
Pablo Escobar	C52342	284.47
Pablo Escobar	Y17133	14.71
Pablo Escobar	NULL	299.18
Roberto	A12347	3284.50
Roberto	B12348	4284.47
Roberto	B22349	5284.29
Roberto	C42340	233.22
Roberto	NULL	13086.48
NULL	NULL	23238.62

Exemplo mais complexo:

```
SELECT cliente.nome, conta.numero , SUM(conta.valor) AS total
FROM conta ,cliente
WHERE conta.codcli = cliente.codigo
GROUP BY cliente.nome, conta.numero WITH ROLLUP;
```

AS – ALL – DISTINCT

AS

Define um nome (“alias”) para uma coluna diferente do rótulo de coluna original

ALL

Especifica que a consulta deverá extrair todos os elementos indicados – é o padrão

DISTINCT

Faz com que o SQL ignore valores repetidos na tabela.

AS

SQL permite dar um nome diferente de uma tabela, campo ou fórmula do nome real existente.

Isso é conseguido com o comando **AS** utilizado junto com o SELECT

Quando pedimos para exibir o nome e a quantidade de contas existentes para cada cliente, foram apresentadas as colunas nome e COUNT. Para que a coluna COUNT seja chamada de contas usamos a seguinte sintaxe:

```
SELECT cliente.nome, COUNT(*) AS contas
FROM cliente, conta
WHERE cliente.codigo = conta.codcli
GROUP BY cliente.nome;
```

nome	contas
Angelo	3
Dedeh	1
Pablo Escobar	2
Roberto	4

DISTINCT

Queremos saber quais as cidades dos nossos clientes, mas como podemos ter vários clientes da mesma cidade, usamos o DISTINCT para não mostrar várias vezes o mesmo nome de cidade.

```
SELECT DISTINCT cidade FROM cliente;
```

```
+----------+
| cidade   |
+----------+
| Pelotas  |
| Campinas |
| Herval   |
+----------+
```

EXERCÍCIOs

1. Apresentar uma listagem identificada pelos apelidos Cliente (para representar o campo nome) e Vencidos (para representar o número de contas vencidas existente na tabela conta que será calculada pela função COUNT) de todos os clientes que possuem contas com vencimento anterior a 31/12/2015.

2. Apresentar uma listagem de contas em atraso, anteriores à data de 31/12/2015, em que devem ser apresentados, além do nome do cliente, o valor da conta, o valor dos juros (10%) e o valor total a ser cobrado, ordenados por cliente.

Subconsulta

Uma subquery é um comando SELECT inserido em uma cláusula de um outro comando SQL.

Pode-se desenvolver comandos sofisticados a partir de comandos simples, utilizando-se subqueries.

Elas podem ser muito úteis quando for necessário selecionar linha a partir de uma tabela com uma condição que dependa de dados da própria tabela.

Subconsulta

A subquery geralmente é identificada como um comando aninhado SELECT.

Em geral, ela é executada primeiro e seu resultado é usado para completar a condição de pesquisa para a pesquisa primária ou externa.

Subconsulta

A subquery deve ser colocada entre parênteses, deve ser colocada depois de um operador de comparação, cláusula ORDER BY não deve ser incluída em uma subquery.

Exemplo:

```
SELECT nome
FROM cliente
WHERE cidade = (SELECT cidade FROM cliente WHERE
nome='Gladislau');
```

nome
Dedeh
Angelo
Roberto
Gladislau
Pablo Escobar

Obs.: Primeiro descobrirá a cidade de Gladislau, depois exibirá o nome dos clientes da mesma cidade que Gladislau.