

Structured Query Language

Prof Eduardo Falcão

eduardo@dca.ufrn.br

FUNCIONARIO

Pnome	Minicial	Unome	<u>Cpf</u>	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr
-------	----------	-------	------------	----------	----------	------	---------	----------------	-----

DEPARTAMENTO

Dnome	<u>Dnumero</u>	Cpf_gerente	Data_inicio_gerente
-------	----------------	-------------	---------------------

LOCALIZACAO_DEP

<u>Dnumero</u>	<u>Dlocal</u>
----------------	---------------

PROJETO

Projnome	<u>Projnumero</u>	Projlocal	Dnum
----------	-------------------	-----------	------

TRABALHA_EM

Fcpf	<u>Pnr</u>	Horas
------	------------	-------

DEPENDENTE

<u>Fcpf</u>	<u>Nome_dependente</u>	Sexo	Datanasc	Parentesco
-------------	------------------------	------	----------	------------

SQL

Structured Query Language

- Suporta DDL e DML
 - DDL: CREATE, ALTER, DROP,
 - DML: INSERT, UPDATE, DELETE
- Permite definir visões sobre o banco de dados, para especificar segurança e autorização, para definir restrições de integridade e para especificar controles de transação
- Não é uma linguagem case-sensitive
- Se baseia no modelo relacional e álgebra relacional

SQL

Structured Query Language

- Originalmente, SQL era chamada de SEQUEL (Structured English QUery Language) e foi criada e implementada na IBM Research (1974)
- Órgãos como ANSI e ISO adotaram a SQL como o padrão oficial de linguagem em ambiente relacional
- O ANSI publicou as padronizações SQL ANSI-89 e ANSI-92
- Revisões da SQL: SQL99 (SQL 3), SQL 2003, e SQL:2006



<https://mariadb.com/downloads/>

DDL

CREATE, ALTER, DROP

CREATE

- Permite criar objetos de BD:
 - CREATE SCHEMA
 - CREATE DOMAIN
 - CREATE TABLE
 - CREATE INDEX
 - CREATE VIEW
 - CREATE TRIGGER
 - ...

CREATE DOMAIN

- Permite declarar um domínio especificando o seu nome para ser usado junto da criação de atributos.

```
/* não tem no mysql, mas tem no postgres */  
CREATE DOMAIN TIPO_ID AS CHAR(4);  
CREATE DOMAIN D_NUM AS INTEGER CHECK (DNUM > 0 AND DNUM <  
21);  
/* podemos usar o check no mysql */
```


SHOW DATABASES

- Um SGBD pode gerenciar múltiplos BDs simultaneamente.

```
MariaDB [UNIVERSIDADE]> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| test |
| universidade |
+-----+
5 rows in set (0.001 sec)
```

CREATE DATABASE

- Crie um BD para seu uso e liste as tabelas para verificar que este BD foi recém criado.

```
CREATE DATABASE UNIVERSIDADE;  
USE UNIVERSIDADE;  
SHOW TABLES;
```

```
MariaDB [mysql]> CREATE DATABASE UNIVERSIDADE;  
Query OK, 1 row affected (0.001 sec)  
  
MariaDB [mysql]> USE UNIVERSIDADE;  
Database changed  
MariaDB [UNIVERSIDADE]> SHOW TABLES;  
Empty set (0.000 sec)  
  
MariaDB [UNIVERSIDADE]>
```

CREATE TABLE

- Usado para criar as relações do BD, especificando o nome da relação, a lista de atributos e seus respectivos domínios, além de outras restrições.

```
CREATE TABLE FUNCIONARIO
(
    atributo1,
    atributo2,
    ...,
    restrição1,
    retrição2,
    ...,
);
```

TIPOS

- <https://mariadb.com/kb/en/data-types/>

FUNCIONARIO

Pnome	Minicial	Unome	<u>Cpf</u>	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr
-------	----------	-------	------------	----------	----------	------	---------	----------------	-----

DEPARTAMENTO

Dnome	<u>Dnumero</u>	Cpf_gerente	Data_inicio_gerente
-------	----------------	-------------	---------------------

LOCALIZACAO_DEP

<u>Dnumero</u>	<u>Dlocal</u>
----------------	---------------

PROJETO

Projnome	<u>Projnumero</u>	Projlocal	Dnum
----------	-------------------	-----------	------

TRABALHA_EM

Fcpf	<u>Pnr</u>	Horas
------	------------	-------

DEPENDENTE

<u>Fcpf</u>	<u>Nome_dependente</u>	Sexo	Datanasc	Parentesco
-------------	------------------------	------	----------	------------

CREATE TABLE

```
CREATE TABLE IF NOT EXISTS FUNCIONARIO(  
    cpf INT PRIMARY KEY,  
    p_nome VARCHAR(50) NOT NULL,  
    m_inicial CHAR,  
    u_nome VARCHAR(50) NOT NULL,  
    dt_nasc DATE NOT NULL,  
    endereco VARCHAR(100),  
    sexo CHAR NOT NULL,  
    salario FLOAT,  
    cpf_sup INT,  
    FOREIGN KEY (cpf_sup) REFERENCES FUNCIONARIO(cpf)  
);
```

CREATE TABLE

```
CREATE TABLE IF NOT EXISTS FUNCIONARIO(  
    cpf INT,  
    p_nome VARCHAR(50) NOT NULL,  
    m_inicial CHAR,  
    u_nome VARCHAR(50) NOT NULL,  
    dt_nasc DATE NOT NULL,  
    endereco VARCHAR(100),  
    sexo CHAR NOT NULL,  
    salario FLOAT,  
    cpf_sup INT,  
    PRIMARY KEY(cpf),  
    FOREIGN KEY (cpf_sup) REFERENCES FUNCIONARIO(cpf)  
);
```

CREATE TABLE

```
CREATE TABLE IF NOT EXISTS FUNCIONARIO(  
    cpf INT PRIMARY KEY,  
    p_nome VARCHAR(50) NOT NULL,  
    m_inicial CHAR,  
    u_nome VARCHAR(50) NOT NULL,  
    dt_nasc DATE NOT NULL,  
    endereco VARCHAR(100),  
    sexo CHAR NOT NULL CHECK(sexo='M' OR sexo="F"),  
    salario FLOAT,  
    cpf_sup INT,  
    FOREIGN KEY (cpf_sup) REFERENCES FUNCIONARIO(cpf)  
);
```


CREATE TABLE

tipo (restrição de domínio)

nome do objeto

```
CREATE TABLE IF NOT EXISTS FUNCIONARIO (
```

```
  cpf INT PRIMARY KEY,
```

```
  p_nome VARCHAR(50) NOT NULL,
```

```
  m_inicial CHAR,
```

```
  u_nome VARCHAR(50) NOT NULL,
```

```
  dt_nasc DATE NOT NULL,
```

```
  endereco VARCHAR(100),
```

```
  sexo CHAR NOT NULL CHECK(sexo='M' OR sexo='F'),
```

```
  salario FLOAT,
```

```
  cpf_sup INT,
```

```
  FOREIGN KEY (cpf_sup) REFERENCES FUNCIONARIO (cpf)
```

```
);
```

restrição de chave

restrição para valores nulos

DESCRIBE <TABLE>

```
MariaDB [universidade]> CREATE TABLE IF NOT EXISTS funcionario(  
-> cpf INT PRIMARY KEY,  
-> p_nome VARCHAR(50) NOT NULL,  
-> m_inicial CHAR,  
-> u_nome VARCHAR(50) NOT NULL,  
-> dt_nasc DATE NOT NULL,  
-> endereco VARCHAR(100),  
-> sexo CHAR NOT NULL CHECK(sexo='M' OR sexo="F"),  
-> salario FLOAT,  
-> cpf_sup INT,  
-> FOREIGN KEY (cpf_sup) REFERENCES Funcionario(cpf)  
-> );
```

Query OK, 0 rows affected (0.047 sec)

```
MariaDB [universidade]> describe funcionario;
```

Field	Type	Null	Key	Default	Extra
cpf	int(11)	NO	PRI	NULL	
p_nome	varchar(50)	NO		NULL	
m_inicial	char(1)	YES		NULL	
u_nome	varchar(50)	NO		NULL	
dt_nasc	date	NO		NULL	
endereco	varchar(100)	YES		NULL	
sexo	char(1)	NO		NULL	
salario	float	YES		NULL	
cpf_sup	int(11)	YES	MUL	NULL	

9 rows in set (0.060 sec)

FUNCIONARIO

Pnome	Minicial	Unome	<u>Cpf</u>	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr
-------	----------	-------	------------	----------	----------	------	---------	----------------	-----

DEPARTAMENTO

Dnome	<u>Dnumero</u>	Cpf_gerente	Data_inicio_gerente
-------	----------------	-------------	---------------------

LOCALIZACAO_DEP

<u>Dnumero</u>	<u>Dlocal</u>
----------------	---------------

PROJETO

Projnome	<u>Projnumero</u>	Projlocal	Dnum
----------	-------------------	-----------	------

TRABALHA_EM

Fcpf	<u>Pnr</u>	Horas
------	------------	-------

DEPENDENTE

<u>Fcpf</u>	<u>Nome_dependente</u>	Sexo	Datanasc	Parentesco
-------------	------------------------	------	----------	------------

Chaves Estrangeiras

```
CREATE TABLE IF NOT EXISTS FUNCIONARIO(  
    cpf INT PRIMARY KEY,  
    p_nome VARCHAR(50) NOT NULL,  
    m_inicial CHAR,  
    u_nome VARCHAR(50) NOT NULL,  
    dt_nasc DATE NOT NULL,  
    endereco VARCHAR(100),  
    sexo CHAR NOT NULL CHECK (sexo='M' OR sexo="F"),  
    salario FLOAT,  
    cpf_sup INT,  
    d_numero INT NOT NULL,  
    FOREIGN KEY (cpf_sup) REFERENCES FUNCIONARIO(cpf),  
    FOREIGN KEY (d_numero) REFERENCES DEPARTAMENTO(d_numero)  
);
```

ALTER TABLE

- Usado para alterações nas definições dos objetos criados no banco de dados.

```
ALTER TABLE FUNCIONARIO  
    especificações de alterações ... ;
```

- O comando ALTER pode ser usado com outros objetos.

```
CREATE TABLE IF NOT EXISTS FUNCIONARIO(  
    cpf INT PRIMARY KEY,  
    ...,  
    d_num INT NOT NULL,  
    FOREIGN KEY (cpf_sup) REFERENCES FUNCIONARIO(cpf)  
);
```

```
CREATE TABLE IF NOT EXISTS DEPARTAMENTO(  
    numero SMALLINT PRIMARY KEY,  
    nome VARCHAR(100),  
    dt_inicio_gerente DATE,  
    cpf_gerente INT NOT NULL,  
    FOREIGN KEY (cpf_gerente) REFERENCES FUNCIONARIO(cpf)  
);
```

```
ALTER TABLE FUNCIONARIO ADD CONSTRAINT FOREIGN KEY (d_num)  
REFERENCES DEPARTAMENTO(numero);
```

Outros exemplos de ALTER TABLE

```
ALTER TABLE FUNCIONARIO ALTER cpf_sup SET DEFAULT '000';
```

```
ALTER TABLE FUNCIONARIO DROP endereco CASCADE;
```

```
ALTER TABLE FUNCIONARIO DROP endereco RESTRICT;
```

RESTRICT and CASCADE are allowed to make porting from other database systems easier. In MariaDB, they do nothing. [\[ref\]](#)

DROP TABLE

- Usado para alterações nas definições dos objetos criados no banco de dados.

```
DROP TABLE FUNCIONARIO CASCADE;
```

`RESTRICT` and `CASCADE` are allowed to make porting from other database systems easier. In MariaDB, they do nothing. [\[ref\]](#)

- O comando DROP pode ser usado com outros objetos.
 - Tudo criado com CREATE pode ser excluído com DROP.

DML

INSERT, UPDATE, DELETE

INSERT

- Usado para inserir uma tupla em uma relação

```
INSERT INTO FUNCIONARIO VALUES(1, 'Eduardo', 'L', 'Falcão', '1989-08-04', 'Rua X, Num  
Y, Bairro Z, João Pessoa', 'M', NULL, NULL, 1);
```

```
INSERT INTO FUNCIONARIO  
VALUES(2, 'Maria', NULL, 'Alice', '1990-12-24', NULL, 'F', NULL, NULL, 1);
```

```
INSERT INTO FUNCIONARIO(cpf, p_nome, u_nome, dt_nasc, sexo, d_numero)  
VALUES(3, 'Alex', 'Soares', '2000-11-14', 'M', 1);
```

INSERT

- Usado para inserir uma tupla em uma relação

```
INSERT INTO DEPARTAMENTO (d_numero, d_nome, cpf_gerente)
```

```
SELECT ...
```

```
FROM ...
```

```
GROUP BY ...
```

Consulta SQL para extração da informação existente em outras relações do mesmo BD. Retorna um conjunto de tuplas e cada uma delas será inserida na relação DEPARTAMENTO através do comando INSERT INTO.

UPDATE

- Usado para modificar valores dos atributos de uma ou mais tuplas de uma relação.

```
UPDATE FUNCIONARIO  
    SET p_nome='Bartolomeu', u_nome='Silva'  
    WHERE cpf=4;
```

Não esqueça o WHERE!!!!

UPDATE

```
UPDATE FUNCIONARIO  
    SET salario=salario*1.1  
    WHERE d_num IN(  
        SELECT ...  
        FROM ...  
        GROUP BY ...);
```

Consulta SQL para extração da informação existente em outras relações do mesmo BD. Retorna um conjunto de tuplas e cada uma delas terá o salário aumentado em 10%.

DELETE

- Usado para remover tuplas de uma relação

```
DELETE FROM FUNCIONARIO;
```

```
DELETE FROM FUNCIONARIO  
WHERE cpf=4;
```

```
DELETE FROM FUNCIONARIO  
WHERE p_nome='Bartolomeu';
```

DELETE

```
DELETE FROM FUNCIONARIO
```

```
WHERE d num IN (
```

```
SELECT ...
```

```
FROM ...
```

```
GROUP BY ...) ;
```

Consulta SQL que retorna um conjunto de tuplas que pertençam àqueles números de departamento, e cada uma delas será removida da relação FUNCIONÁRIO.

Atividade:
criar demais
tabelas
especificadas
no seguinte
modelo

FUNCIONARIO

Pnome	Minicial	Unome	<u>Cpf</u>	Datanasc	Endereco	Sexo	Salario	Cpf_supervisor	Dnr
-------	----------	-------	------------	----------	----------	------	---------	----------------	-----

DEPARTAMENTO

Dnome	<u>Dnumero</u>	Cpf_gerente	Data_inicio_gerente
-------	----------------	-------------	---------------------

LOCALIZACAO_DEP

<u>Dnumero</u>	<u>Dlocal</u>
----------------	---------------

PROJETO

Projnome	<u>Projnumero</u>	Projlocal	Dnum
----------	-------------------	-----------	------

TRABALHA_EM

<u>Fcpf</u>	<u>Pnr</u>	Horas
-------------	------------	-------

DEPENDENTE

<u>Fcpf</u>	<u>Nome_dependente</u>	Sexo	Datanasc	Parentesco
-------------	------------------------	------	----------	------------

RECUPERAÇÃO

SELECT

SELECT

- Usado para extração de informação.
 - Estrutura básica:

```
SELECT <lista de atributos>  
FROM <lista de tabelas>  
WHERE <condição>
```

SELECT

```
SELECT * FROM FUNCIONARIO;
```

```
SELECT * FROM FUNCIONARIO WHERE sexo='M';
```

```
SELECT p_nome, u_nome FROM FUNCIONARIO  
WHERE sexo='M';
```

```
SELECT p_nome, u_nome FROM FUNCIONARIO  
WHERE sexo='M' AND d_numero=1;
```

SELECT

```
SELECT * FROM FUNCIONARIO;
```

```
SELECT * FROM FUNCIONARIO WHERE sexo='M';
```

```
SELECT p_nome, u_nome FROM FUNCIONARIO  
WHERE sexo='M';
```

```
SELECT p_nome, u_nome FROM FUNCIONARIO  
WHERE sexo='M' AND d_numero=1;
```

SELECT

SELECT com duas tabelas resulta em um produto cartesiano.

```
SELECT FUNCIONARIO.p_nome, FUNCIONARIO.u_nome, DEPARTAMENTO.d_nome  
FROM FUNCIONARIO, DEPARTAMENTO;
```

p_nome	u_nome	d_nome
Eduardo	Falcão	Departamento de Engenharia de Computação e Automação
Eduardo	Falcão	Departamento de Informática e Matemática Aplicada
Maria	Alice	Departamento de Engenharia de Computação e Automação
Maria	Alice	Departamento de Informática e Matemática Aplicada
Alex	Soares	Departamento de Engenharia de Computação e Automação
Alex	Soares	Departamento de Informática e Matemática Aplicada
Bartolomeu	Silva	Departamento de Engenharia de Computação e Automação
Bartolomeu	Silva	Departamento de Informática e Matemática Aplicada

SELECT

SELECT com duas tabelas resulta em um produto cartesiano.

```
SELECT FUNCIONARIO.p_nome, FUNCIONARIO.u_nome, DEPARTAMENTO.d_nome  
FROM FUNCIONARIO, DEPARTAMENTO  
WHERE sexo='M';
```

p_nome	u_nome	d_nome
Eduardo	Falcão	Departamento de Engenharia de Computação e Automação
Eduardo	Falcão	Departamento de Informática e Matemática Aplicada
Alex	Soares	Departamento de Engenharia de Computação e Automação
Alex	Soares	Departamento de Informática e Matemática Aplicada
Bartolomeu	Silva	Departamento de Engenharia de Computação e Automação
Bartolomeu	Silva	Departamento de Informática e Matemática Aplicada

SELECT

SELECT com duas tabelas resulta em um produto cartesiano.

```
SELECT FUNCIONARIO.p_nome, FUNCIONARIO.u_nome, DEPARTAMENTO.d_nome  
FROM FUNCIONARIO, DEPARTAMENTO  
WHERE sexo='M' AND FUNCIONARIO.d_numero=DEPARTAMENTO.d_numero;
```

p_nome	u_nome	d_nome
Eduardo	Falcão	Departamento de Engenharia de Computação e Automação
Alex	Soares	Departamento de Informática e Matemática Aplicada
Bartolomeu	Silva	Departamento de Informática e Matemática Aplicada

SELECT

SELECT com duas tabelas resulta em um produto cartesiano.

```
SELECT FUNCIONARIO.p_nome, FUNCIONARIO.u_nome, DEPARTAMENTO.d_nome  
FROM FUNCIONARIO, DEPARTAMENTO  
WHERE sexo='M' AND FUNCIONARIO.d_numero=DEPARTAMENTO.d_numero AND  
DEPARTAMENTO.d_numero=1;
```

p_nome	u_nome	d_nome
Eduardo	Falcão	Departamento de Engenharia de Computação e Automação

SELECT

Para cada projeto localizado em 'Natal', liste o nome do projeto, o nome do departamento que o controla, e o nome endereço e data de nascimento do gerente desse departamento

```
SELECT PROJETO.p_nome, DEPARTAMENTO.d_nome, FUNCIONARIO.p_nome,  
       FUNCIONARIO.u_nome, FUNCIONARIO.endereco, FUNCIONARIO.dt_nasc  
FROM PROJETO, DEPARTAMENTO, FUNCIONARIO;
```

```
SELECT PROJETO.p_nome, DEPARTAMENTO.d_nome,  
       FUNCIONARIO.p_nome, FUNCIONARIO.u_nome,  
       FUNCIONARIO.endereco, FUNCIONARIO.dt_nasc  
FROM PROJETO, DEPARTAMENTO, FUNCIONARIO;
```

Para cada projeto localizado em 'Natal', liste o nome do projeto, o nome do departamento que o controla, e o nome endereço e data de nascimento do gerente desse departamento

p_nome	d_nome	p_nome	u_nome	endereco	dt_nasc
Segurança em IoT	Departamento de Engenharia de Computação e Automação	Eduardo	Falcão	Rua X, Num Y, Bairro Z, João Pessoa	1989-08-04
Segurança em IoT	Departamento de Informática e Matemática Aplicada	Eduardo	Falcão	Rua X, Num Y, Bairro Z, João Pessoa	1989-08-04
Robótica Educacional	Departamento de Engenharia de Computação e Automação	Eduardo	Falcão	Rua X, Num Y, Bairro Z, João Pessoa	1989-08-04
Robótica Educacional	Departamento de Informática e Matemática Aplicada	Eduardo	Falcão	Rua X, Num Y, Bairro Z, João Pessoa	1989-08-04
Blockchain	Departamento de Engenharia de Computação e Automação	Eduardo	Falcão	Rua X, Num Y, Bairro Z, João Pessoa	1989-08-04
Blockchain	Departamento de Informática e Matemática Aplicada	Eduardo	Falcão	Rua X, Num Y, Bairro Z, João Pessoa	1989-08-04
Segurança em IoT	Departamento de Engenharia de Computação e Automação	Maria	Alice	NULL	1990-12-24
Segurança em IoT	Departamento de Informática e Matemática Aplicada	Maria	Alice	NULL	1990-12-24
Robótica Educacional	Departamento de Engenharia de Computação e Automação	Maria	Alice	NULL	1990-12-24
Robótica Educacional	Departamento de Informática e Matemática Aplicada	Maria	Alice	NULL	1990-12-24
Blockchain	Departamento de Engenharia de Computação e Automação	Maria	Alice	NULL	1990-12-24
Blockchain	Departamento de Informática e Matemática Aplicada	Maria	Alice	NULL	1990-12-24
Segurança em IoT	Departamento de Engenharia de Computação e Automação	Alex	Soares	NULL	2000-11-14
Segurança em IoT	Departamento de Informática e Matemática Aplicada	Alex	Soares	NULL	2000-11-14
Robótica Educacional	Departamento de Engenharia de Computação e Automação	Alex	Soares	NULL	2000-11-14
Robótica Educacional	Departamento de Informática e Matemática Aplicada	Alex	Soares	NULL	2000-11-14
Blockchain	Departamento de Engenharia de Computação e Automação	Alex	Soares	NULL	2000-11-14
Blockchain	Departamento de Informática e Matemática Aplicada	Alex	Soares	NULL	2000-11-14
Segurança em IoT	Departamento de Engenharia de Computação e Automação	Bartolomeu	Silva	NULL	1970-12-24
Segurança em IoT	Departamento de Informática e Matemática Aplicada	Bartolomeu	Silva	NULL	1970-12-24
Robótica Educacional	Departamento de Engenharia de Computação e Automação	Bartolomeu	Silva	NULL	1970-12-24
Robótica Educacional	Departamento de Informática e Matemática Aplicada	Bartolomeu	Silva	NULL	1970-12-24
Blockchain	Departamento de Engenharia de Computação e Automação	Bartolomeu	Silva	NULL	1970-12-24
Blockchain	Departamento de Informática e Matemática Aplicada	Bartolomeu	Silva	NULL	1970-12-24

Para cada projeto localizado em 'Natal', liste o nome do projeto, o nome do departamento que o controla, e o nome endereço e data de nascimento do gerente desse departamento

```
SELECT PROJETO.p_nome, DEPARTAMENTO.d_nome, FUNCIONARIO.p_nome,
       FUNCIONARIO.u_nome, FUNCIONARIO.endereco, FUNCIONARIO.dt_nasc
FROM PROJETO, DEPARTAMENTO, FUNCIONARIO
WHERE PROJETO.p_local='Natal' AND
       PROJETO.d_numero=DEPARTAMENTO.d_numeroAND
       DEPARTAMENTO.f_cpf_gerente=FUNCIONARIO.cpf;
```

p_nome	d_nome	p_nome	u_nome	endereco	dt_nasc
Segurança em IoT	Departamento de Engenharia de Computação e Automação	Eduardo	Falcão	Rua X, Num Y, Bairro Z, João Pessoa	1989-08-04
Robótica Educacional	Departamento de Engenharia de Computação e Automação	Eduardo	Falcão	Rua X, Num Y, Bairro Z, João Pessoa	1989-08-04
Blockchain	Departamento de Informática e Matemática Aplicada	Maria	Alice	NULL	1990-12-24

SELECT

Para cada funcionário, recupere seu nome e o nome do seu supervisor direto.

```
UPDATE FUNCIONARIO
```

```
    SET cpf_sup=1
```

```
    WHERE cpf=2;
```

```
UPDATE FUNCIONARIO
```

```
    SET cpf_sup=2
```

```
    WHERE cpf=3;
```

```
UPDATE FUNCIONARIO
```


```
    SET cpf_sup=2
```

```
    WHERE cpf=4;
```

SELECT

Para cada funcionário, recupere seu nome e o nome do seu supervisor direto.

```
SELECT F.p_nome, F.u_nome, F.cpf, F.cpf_sup,  
       S.p_nome, S.u_nome, S.cpf, S.cpf_sup  
FROM FUNCIONARIO as F, FUNCIONARIO as S;
```



p_nome	u_nome	cpf	cpf_sup	p_nome	u_nome	cpf	cpf_sup
Eduardo	Falcão	1	NULL	Eduardo	Falcão	1	NULL
Maria	Alice	2	1	Eduardo	Falcão	1	NULL
Alex	Soares	3	2	Eduardo	Falcão	1	NULL
Bartolomeu	Silva	4	2	Eduardo	Falcão	1	NULL
Eduardo	Falcão	1	NULL	Maria	Alice	2	1
Maria	Alice	2	1	Maria	Alice	2	1
Alex	Soares	3	2	Maria	Alice	2	1
Bartolomeu	Silva	4	2	Maria	Alice	2	1
Eduardo	Falcão	1	NULL	Alex	Soares	3	2
Maria	Alice	2	1	Alex	Soares	3	2
Alex	Soares	3	2	Alex	Soares	3	2
Bartolomeu	Silva	4	2	Alex	Soares	3	2
Eduardo	Falcão	1	NULL	Bartolomeu	Silva	4	2
Maria	Alice	2	1	Bartolomeu	Silva	4	2
Alex	Soares	3	2	Bartolomeu	Silva	4	2
Bartolomeu	Silva	4	2	Bartolomeu	Silva	4	2

SELECT

Para cada funcionário, recupere seu nome e o nome do seu supervisor direto.

```
SELECT F.p_nome, F.u_nome, S.p_nome, S.u_nome
      FROM FUNCIONARIO as F, FUNCIONARIO as S
      WHERE F.cpf_sup=S.cpf;
```

p_nome	u_nome	p_nome	u_nome
Maria	Alice	Eduardo	Falcão
Alex	Soares	Maria	Alice
Bartolomeu	Silva	Maria	Alice

SELECT

Recupere o salário de todos os funcionários.

```
UPDATE FUNCIONARIO  
    SET salario=5000  
    WHERE cpf=1;
```

```
UPDATE FUNCIONARIO  
    SET salario=8000  
    WHERE cpf=2;
```

```
UPDATE FUNCIONARIO  
    SET salario=3000  
    WHERE cpf=3;
```

```
UPDATE FUNCIONARIO  
    SET salario=5000  
    WHERE cpf=4;
```

SELECT

Recupere o salário de todos os funcionários.

```
SELECT ALL salario FROM FUNCIONARIO;
```

```
MariaDB [UNIVERSIDADE]> SELECT ALL salario FROM FUNCIONARIO;
```

salario
5000
8000
3000
5000

```
4 rows in set (0.000 sec)
```


SELECT

Recupere os valores distintos de salário praticados na universidade.

```
SELECT DISTINCT salario FROM FUNCIONARIO;
```

```
MariaDB [UNIVERSIDADE]> SELECT DISTINCT salario FROM FUNCIONARIO;
```

```
+-----+  
| salario |  
+-----+  
|    5000 |  
|    8000 |  
|    3000 |  
+-----+
```

```
3 rows in set (0.003 sec)
```

SELECT

```
/* Faça uma lista de números de projetos que envolvem um funcionário  
* cujo sobrenome é 'Silva'. O envolvimento do funcionário pode ser  
* como trabalhador ou como gerente do departamento que controla o  
* projeto.  
*/
```

```
/* alocando Bartolomeu Silva (cpf: 4) para Projeto 1*/  
INSERT INTO TRABALHA_EM(f_cpf,p_numero,horas)  
VALUES (4,1,20);  
  
/* alocando Maria Silva (cpf: 4) para Projeto 1*/  
INSERT INTO FUNCIONARIO(cpf,p_nome,u_nome,dt_nasc,sexo,d_numero,salario)  
VALUES (5,'Maria','Silva','1980-12-10','F',2,9000);  
INSERT INTO TRABALHA_EM(f_cpf,p_numero,horas)  
VALUES (5,1,40);
```

SELECT

```
/* Faça uma lista de números de projetos que envolvem um funcionário  
* cujo sobrenome é 'Silva'. O envolvimento do funcionário pode ser  
* como trabalhador ou como gerente do departamento que controla o  
* projeto.  
*/
```

```
SELECT PROJETO.p_nome  
      FROM FUNCIONARIO, TRABALHA_EM, PROJETO  
     WHERE PROJETO.p_numero=TRABALHA_EM.p_numero AND  
           TRABALHA_EM.f_cpf=FUNCIONARIO.cpf;
```

```
MariaDB [UNIVERSIDADE]> SELECT PROJETO.p_nome  
->      FROM FUNCIONARIO, TRABALHA_EM, PROJETO  
->      WHERE PROJETO.p_numero=TRABALHA_EM.p_numero AND  
->            TRABALHA_EM.f_cpf=FUNCIONARIO.cpf;  
+-----+  
| p_nome          |  
+-----+  
| Segurança em IoT |  
| Segurança em IoT |  
+-----+  
2 rows in set (0.001 sec)
```

SELECT

```
/* Faça uma lista de números de projetos que envolvem um funcionário  
* cujo sobrenome é 'Silva'. O envolvimento do funcionário pode ser  
* como trabalhador ou como gerente do departamento que controla o  
* projeto.  
*/
```

```
SELECT DISTINCT PROJETO.p_nome  
FROM FUNCIONARIO, TRABALHA_EM, PROJETO  
WHERE PROJETO.p_numero=TRABALHA_EM.p_numero AND  
TRABALHA_EM.f_cpf=FUNCIONARIO.cpf;
```

```
MariaDB [UNIVERSIDADE]> SELECT DISTINCT PROJETO.p_nome  
-> FROM FUNCIONARIO, TRABALHA_EM, PROJETO  
-> WHERE PROJETO.p_numero=TRABALHA_EM.p_numero AND  
-> TRABALHA_EM.f_cpf=FUNCIONARIO.cpf;  
+-----+  
| p_nome |  
+-----+  
| Segurança em IoT |  
+-----+  
1 row in set (0.001 sec)
```

SELECT

```
/* Faça uma lista de números de projetos que envolvem um funcionário  
* cujo sobrenome é 'Silva'. O envolvimento do funcionário pode ser  
* como trabalhador ou como gerente do departamento que controla o  
* projeto.  
*/
```

```
/* alocando Joana Silva (cpf: 6) para Projeto 2 como gerente do  
departamento*/
```

```
INSERT INTO FUNCIONARIO(cpf,p_nome,u_nome,dt_nasc,sexo,d_numero,salario)  
VALUES(6,'Joana','Silva','1985-02-10','F',2,13000);
```

```
INSERT INTO TRABALHA_EM(f_cpf,p_numero,horas)  
VALUES(6,3,40);
```

```
UPDATE DEPARTAMENTO  
SET f_cpf_gerente=6  
WHERE D_NUMERO=2;
```

SELECT

```
/* Faça uma lista de números de projetos que envolvem um funcionário
 * cujo sobrenome é 'Silva'. O envolvimento do funcionário pode ser
 * como trabalhador ou como gerente do departamento que controla o
 * projeto.
 */
```

```
SELECT DISTINCT PROJETO.p_nome
FROM FUNCIONARIO, TRABALHA_EM, PROJETO, DEPARTAMENTO
WHERE PROJETO.p_numero=TRABALHA_EM.p_numero AND
      TRABALHA_EM.f_cpf=FUNCIONARIO.cpf AND
      DEPARTAMENTO.f_cpf_gerente=FUNCIONARIO.cpf;
```

```
MariaDB [UNIVERSIDADE]> SELECT DISTINCT PROJETO.p_nome
-> FROM FUNCIONARIO, TRABALHA_EM, PROJETO, DEPARTAMENTO
-> WHERE PROJETO.p_numero=TRABALHA_EM.p_numero AND
-> TRABALHA_EM.f_cpf=FUNCIONARIO.cpf AND
-> DEPARTAMENTO.f_cpf_gerente=FUNCIONARIO.cpf;
+-----+
| p_nome |
+-----+
| Blockchain |
+-----+
1 row in set (0.001 sec)
```

```

/* Faça uma lista de números de projetos que envolvem um funcionário
 * cujo sobrenome é 'Silva'. O envolvimento do funcionário pode ser
 * como trabalhador ou como gerente do departamento que controla o
 * projeto.
 */

```

```

(SELECT DISTINCT PROJETO.p_nome
 FROM FUNCIONARIO, TRABALHA_EM, PROJETO
 WHERE PROJETO.p_numero=TRABALHA_EM.p_numero AND
        TRABALHA_EM.f_cpf=FUNCIONARIO.cpf)
UNION
(SELECT DISTINCT PROJETO.p_nome
 FROM FUNCIONARIO, TRABALHA_EM, PROJETO, DEPARTAMENTO
 WHERE PROJETO.p_numero=TRABALHA_EM.p_numero AND
        TRABALHA_EM.f_cpf=FUNCIONARIO.cpf AND
        DEPARTAMENTO.f_cpf_gerente=FUNCIONARIO.cpf);

```

```

MariaDB [UNIVERSIDADE]> (SELECT DISTINCT PROJETO.p_nome
-> FROM FUNCIONARIO, TRABALHA_EM, PROJETO
-> WHERE PROJETO.p_numero=TRABALHA_EM.p_numero AND
-> TRABALHA_EM.f_cpf=FUNCIONARIO.cpf)
-> UNION
-> (SELECT DISTINCT PROJETO.p_nome
-> FROM FUNCIONARIO, TRABALHA_EM, PROJETO, DEPARTAMENTO
-> WHERE PROJETO.p_numero=TRABALHA_EM.p_numero AND
-> TRABALHA_EM.f_cpf=FUNCIONARIO.cpf AND
-> DEPARTAMENTO.f_cpf_gerente=FUNCIONARIO.cpf);
+-----+
| p_nome |
+-----+
| Segurança em IoT |
| Blockchain |
+-----+
2 rows in set (0.001 sec)

```

```
(SELECT DISTINCT PROJETO.p_nome
  FROM FUNCIONARIO, TRABALHA_EM, PROJETO
 WHERE PROJETO.p_numero=TRABALHA_EM.p_numero AND
        TRABALHA_EM.f_cpf=FUNCIONARIO.cpf)
UNION
(SELECT DISTINCT PROJETO.p_nome
  FROM FUNCIONARIO, TRABALHA_EM, PROJETO, DEPARTAMENTO
 WHERE PROJETO.p_numero=TRABALHA_EM.p_numero AND
        TRABALHA_EM.f_cpf=FUNCIONARIO.cpf AND
        DEPARTAMENTO.f_cpf_gerente=FUNCIONARIO.cpf);
```

Tuplas em duplicidade serão eliminadas do resultado,
pois se trata de operações sobre conjuntos.

Use UNION/EXCEPT/INTERCEPT **ALL** se for
necessário manter as duplicatas.


```
/* Recupere todos os funcionários cujos endereços são em JP */
```

SELECT

```
UPDATE FUNCIONARIO
```

```
  SET endereco='Rua X, Num Y, Bairro Z, JP'
```

```
  WHERE cpf=1;
```

```
UPDATE FUNCIONARIO
```

```
  SET endereco='Rua X, Num Y, Bairro Z, Natal'
```

```
  WHERE cpf=2;
```

```
UPDATE FUNCIONARIO
```

```
  SET endereco='Rua X, Num Y, Bairro Z, CG'
```

```
  WHERE cpf=3;
```

```
UPDATE FUNCIONARIO
```

```
  SET endereco='Rua X, Num Y, Bairro Z, JP'
```

```
  WHERE cpf=4;
```

```
UPDATE FUNCIONARIO
```

```
  SET endereco='Rua X, Num Y, Bairro Z, Natal'
```

```
  WHERE cpf=5;
```

```
UPDATE FUNCIONARIO
```

```
  SET endereco='Rua X, Num Y, Bairro Z, CG'
```

```
  WHERE cpf=6;
```

SELECT

/ Recupere todos os funcionários cujos endereços são em JP */*

```
SELECT p_nome, u_nome  
FROM FUNCIONARIO  
WHERE endereco LIKE '%JP%';
```

```
MariaDB [UNIVERSIDADE]> SELECT p_nome, u_nome  
-> FROM FUNCIONARIO  
-> WHERE endereco LIKE '%JP%';  
+-----+-----+  
| p_nome | u_nome |  
+-----+-----+  
| Eduardo | Falcão |  
| Bartolomeu | Silva |  
+-----+-----+  
2 rows in set (0.043 sec)
```

SELECT

```
/* Recupere todos os funcionários que nasceram na década de 80 */
```

```
SELECT p_nome, u_nome  
      FROM FUNCIONARIO  
     WHERE dt_nasc LIKE '198_--_--';
```

```
MariaDB [UNIVERSIDADE]> SELECT p_nome, u_nome  
->      FROM FUNCIONARIO  
->      WHERE dt_nasc LIKE '198_--_--';  
+-----+-----+  
| p_nome | u_nome |  
+-----+-----+  
| Eduardo | Falcão |  
| Maria   | Silva  |  
| Joana   | Silva  |  
+-----+-----+  
3 rows in set (0.000 sec)
```

SELECT

```
/* Recuperar a lista de funcionários e de projetos nos quais eles
* trabalham, ordenada por departamento. Dentro de departamento,
* ordene por nome do funcionário.
*/
```

```
SELECT D.d_nome, F.p_nome, P.p_nome
FROM DEPARTAMENTO AS D,
     FUNCIONARIO AS F,
     PROJETO AS P,
     TRABALHA_EM AS T
WHERE P.p_numero=T.p_numero AND
     T.f_cpf=F.cpf AND
     F.d_numero = D.d_numero;
```

```
MariaDB [UNIVERSIDADE]> SELECT D.d_nome, F.p_nome, P.p_nome
```

```
-> FROM DEPARTAMENTO AS D,
->      FUNCIONARIO AS F,
->      PROJETO AS P,
->      TRABALHA_EM AS T
-> WHERE P.p_numero=T.p_numero AND
->      T.f_cpf=F.cpf AND
->      F.d_numero = D.d_numero;
```

d_nome	p_nome	p_nome
Departamento de Informática e Matemática Aplicada	Bartolomeu	Segurança em IoT
Departamento de Informática e Matemática Aplicada	Maria	Segurança em IoT
Departamento de Informática e Matemática Aplicada	Joana	Blockchain

3 rows in set (0.001 sec)

SELECT

```
/* Recuperar a lista de funcionários e de projetos nos quais eles
* trabalham, ordenada por departamento. Dentro de departamento,
* ordene por nome do funcionário.
*/
```

```
SELECT D.d_nome, F.p_nome, P.p_nome
FROM DEPARTAMENTO AS D,
     FUNCIONARIO AS F,
     PROJETO AS P,
     TRABALHA_EM AS T
WHERE P.p_numero=T.p_numero AND
      T.f_cpf=F.cpf AND
      F.d_numero = D.d_numero
ORDER BY D.d_nome ASC, F.p_nome ASC;
```

```
MariaDB [UNIVERSIDADE]> SELECT D.d_nome, F.p_nome, P.p_nome
-> FROM DEPARTAMENTO AS D,
-> FUNCIONARIO AS F,
-> PROJETO AS P,
-> TRABALHA_EM AS T
-> WHERE P.p_numero=T.p_numero AND
-> T.f_cpf=F.cpf AND
-> F.d_numero = D.d_numero
-> ORDER BY D.d_nome ASC, F.p_nome ASC;
```

d_nome	p_nome	p_nome
Departamento de Informática e Matemática Aplicada	Bartolomeu	Segurança em IoT
Departamento de Informática e Matemática Aplicada	Joana	Blockchain
Departamento de Informática e Matemática Aplicada	Maria	Segurança em IoT

3 rows in set (0.042 sec)

SELECT

```
SELECT p_nome, m_inicial, u_nome
      FROM FUNCIONARIO
      WHERE cpf_sup IS NULL;
```

/* Recupere o nome dos funcionários que
* não possuem supervisores
*/

```
MariaDB [UNIVERSIDADE]> SELECT * FROM FUNCIONARIO;
```

cpf	p_nome	m_inicial	u_nome	dt_nasc	endereco	sexo	salario	cpf_sup	d_numero
1	Eduardo	L	Falcão	1989-08-04	Rua X, Num Y, Bairro Z, JP	M	5000	NULL	1
2	Maria	NULL	Alice	1990-12-24	Rua X, Num Y, Bairro Z, Natal	F	8000	1	1
3	Alex	NULL	Soares	2000-11-14	Rua X, Num Y, Bairro Z, CG	M	3000	2	2
4	Bartolomeu	NULL	Silva	1970-12-24	Rua X, Num Y, Bairro Z, JP	M	5000	2	2
5	Maria	NULL	Silva	1980-12-10	Rua X, Num Y, Bairro Z, Natal	F	9000	NULL	2
6	Joana	NULL	Silva	1985-02-10	Rua X, Num Y, Bairro Z, CG	F	13000	NULL	2

```
6 rows in set (0.001 sec)
```

```
MariaDB [UNIVERSIDADE]> SELECT p_nome, m_inicial, u_nome
-> FROM FUNCIONARIO
-> WHERE cpf_sup IS NULL;
```

p_nome	m_inicial	u_nome
Eduardo	L	Falcão
Maria	NULL	Silva
Joana	NULL	Silva

```
3 rows in set (0.040 sec)
```

SELECT com (INNER) JOIN

/ Recupere o cpf, primeiro nome, e último nome, e nome do departamento que os funcionários trabalham */*

```
SELECT F.cpf, F.p_nome, F.u_nome, D.d_nome
      FROM FUNCIONARIO AS F, DEPARTAMENTO AS D
      WHERE F.d_numero=D.d_numero;
```

```
SELECT F.cpf, F.p_nome, F.u_nome, D.d_nome
      FROM FUNCIONARIO AS F
      JOIN DEPARTAMENTO AS D
      ON F.d_numero=D.d_numero;
```

condição de junção

```
MariaDB [UNIVERSIDADE]> SELECT F.cpf, F.p_nome, F.u_nome, D.d_nome
->      FROM FUNCIONARIO AS F, DEPARTAMENTO AS D
->      WHERE F.d_numero=D.d_numero;
```

cpf	p_nome	u_nome	d_nome
1	Eduardo	Falcão	Departamento de Engenharia de Computação e Automação
2	Maria	Alice	Departamento de Engenharia de Computação e Automação
3	Alex	Soares	Departamento de Informática e Matemática Aplicada
4	Bartolomeu	Silva	Departamento de Informática e Matemática Aplicada
5	Maria	Silva	Departamento de Informática e Matemática Aplicada
6	Joana	Silva	Departamento de Informática e Matemática Aplicada

6 rows in set (0.000 sec)

```
MariaDB [UNIVERSIDADE]> SELECT F.cpf, F.p_nome, F.u_nome, D.d_nome
->      FROM FUNCIONARIO AS F
->      JOIN DEPARTAMENTO AS D
->      ON F.d_numero=D.d_numero;
```

cpf	p_nome	u_nome	d_nome
1	Eduardo	Falcão	Departamento de Engenharia de Computação e Automação
2	Maria	Alice	Departamento de Engenharia de Computação e Automação
3	Alex	Soares	Departamento de Informática e Matemática Aplicada
4	Bartolomeu	Silva	Departamento de Informática e Matemática Aplicada
5	Maria	Silva	Departamento de Informática e Matemática Aplicada
6	Joana	Silva	Departamento de Informática e Matemática Aplicada

6 rows in set (0.000 sec)

SELECT com (INNER) JOIN

Para cada funcionário, recupere seu nome e o nome do seu supervisor direto.

```
SELECT F.p_nome, F.u_nome, S.p_nome, S.u_nome
      FROM FUNCIONARIO as F, FUNCIONARIO as S
      WHERE F.cpf_sup=S.cpf;
```

```
SELECT F.p_nome, F.u_nome, S.p_nome, S.u_nome
      FROM FUNCIONARIO AS F JOIN FUNCIONARIO AS s
      ON F.cpf_sup=S.cpf;
```

condição de junção

```
MariaDB [UNIVERSIDADE]> SELECT F.p_nome, F.u_nome, S.p_nome, S.u_nome
->      FROM FUNCIONARIO as F, FUNCIONARIO as S
->      WHERE F.cpf_sup=S.cpf;
```

p_nome	u_nome	p_nome	u_nome
Maria	Alice	Eduardo	Falcão
Alex	Soares	Maria	Alice
Bartolomeu	Silva	Maria	Alice

3 rows in set (0.000 sec)

```
MariaDB [UNIVERSIDADE]> SELECT F.p_nome, F.u_nome, S.p_nome, S.u_nome
->      FROM FUNCIONARIO AS F JOIN FUNCIONARIO AS s
->      ON F.cpf_sup=S.cpf;
```

p_nome	u_nome	p_nome	u_nome
Maria	Alice	Eduardo	Falcão
Alex	Soares	Maria	Alice
Bartolomeu	Silva	Maria	Alice

3 rows in set (0.000 sec)

Exemplos com Funções Agregadas

```
SELECT SUM(salario), MAX(salario), MIN(salario), AVG(salario)
FROM FUNCIONARIO;
```

```
MariaDB [UNIVERSIDADE]> SELECT SUM(salario), MAX(salario), MIN(salario), AVG(salario)
-> FROM FUNCIONARIO;
```

SUM(salario)	MAX(salario)	MIN(salario)	AVG(salario)
43000	13000	3000	7166.666666666667

```
1 row in set (0.001 sec)
```

Exemplos com Funções Agregadas

```
INSERT INTO DEPENDENTE(f_cpf,nome,sexo,dt_nasc,parentesco)
VALUES(1,'Aline','F','2015-12-12','filha');
INSERT INTO DEPENDENTE(f_cpf,nome,sexo,dt_nasc,parentesco)
VALUES(1,'Marcos','M','2018-10-18','filho');
INSERT INTO DEPENDENTE(f_cpf,nome,sexo,dt_nasc,parentesco)
VALUES(1,'José','M','2010-09-08','sobrinho');
INSERT INTO DEPENDENTE(f_cpf,nome,sexo,dt_nasc,parentesco)
VALUES(2,'Mariane','F','2019-09-23','filha');
```

```
SELECT p_nome
FROM FUNCIONARIO
WHERE (
    SELECT COUNT(*)
    FROM DEPENDENTE
    WHERE cpf=f_cpf) >= 2;
```

```
MariaDB [UNIVERSIDADE]> SELECT p_nome
-> FROM FUNCIONARIO
-> WHERE(
-> SELECT COUNT(*)
-> FROM DEPENDENTE
-> WHERE cpf=f_cpf) >= 2;
```

```
+-----+
| p_nome |
+-----+
| Eduardo |
+-----+
```

```
1 row in set (0.001 sec)
```

Exemplos com GROUP BY e HAVING

```
SELECT F.cpf, F.p_nome, F.u_nome, F.salario, D.d_nome
FROM FUNCIONARIO AS F, DEPARTAMENTO AS D
WHERE F.d_numero=D.d_numero;
```

```
MariaDB [UNIVERSIDADE]> SELECT F.cpf, F.p_nome, F.u_nome, F.salario, D.d_nome
-> FROM FUNCIONARIO AS F, DEPARTAMENTO AS D
-> WHERE F.d_numero=D.d_numero;
```

cpf	p_nome	u_nome	salario	d_nome
1	Eduardo	Falcão	5000	Departamento de Engenharia de Computação e Automação
2	Maria	Alice	8000	Departamento de Engenharia de Computação e Automação
3	Alex	Soares	3000	Departamento de Informática e Matemática Aplicada
4	Bartolomeu	Silva	5000	Departamento de Informática e Matemática Aplicada
5	Maria	Silva	9000	Departamento de Informática e Matemática Aplicada
6	Joana	Silva	13000	Departamento de Informática e Matemática Aplicada

```
6 rows in set (0.001 sec)
```

Exemplos com GROUP BY e HAVING

```
SELECT D.d_nome, COUNT(*)  
FROM DEPARTAMENTO AS D,  
      FUNCIONARIO AS F  
WHERE D.d_numero=F.d_numero AND  
      F.salario >= 5000  
GROUP BY D.d_nome;
```

```
MariaDB [UNIVERSIDADE]> SELECT D.d_nome, COUNT(*)  
-> FROM DEPARTAMENTO AS D,  
->      FUNCIONARIO AS F  
-> WHERE D.d_numero=F.d_numero AND  
->      F.salario >= 5000  
-> GROUP BY D.d_nome;
```

d_nome	COUNT(*)
Departamento de Engenharia de Computação e Automação	2
Departamento de Informática e Matemática Aplicada	3

2 rows in set (0.001 sec)

Exemplos com GROUP BY e HAVING

```
SELECT D.d_nome, COUNT(*)
  FROM DEPARTAMENTO AS D,
       FUNCIONARIO AS F
 WHERE D.d_numero=F.d_numero AND
       F.salario >= 5000
 GROUP BY D.d_nome
 HAVING COUNT(*) > 2;
```

```
MariaDB [UNIVERSIDADE]> SELECT D.d_nome, COUNT(*)
->      FROM DEPARTAMENTO AS D,
->      FUNCIONARIO AS F
->      WHERE D.d_numero=F.d_numero AND
->      F.salario >= 5000
->      GROUP BY D.d_nome
->      HAVING COUNT(*) > 2;
```

d_nome	COUNT(*)
Departamento de Informática e Matemática Aplicada	3

1 row in set (0.000 sec)