

Banco de Dados

MySQL: INSERT e SELECT

Objetivos

- Inserção de dados em tabelas de um banco de dados.
- Realização de consultas em um banco de dados.

- **INSERT**

- O comando **INSERT** é utilizado para inserir novos registros de dados em uma tabela de um banco de dados.
 - Em outras palavras, inserir uma linha na tabela.
- Sintaxe:

```
INSERT INTO <tabela> [lista de colunas] VALUES  
    <lista de valores>,  
    [lista de valores, ...];
```

- **INSERT**

- O comando **INSERT** é utilizado para inserir novos registros de dados em uma tabela de um banco de dados.
 - Em outras palavras, inserir uma linha na tabela.
- Sintaxe:

```
INSERT INTO <tabela> [lista de colunas] VALUES  
    <lista de valores>,  
    [lista de valores, ...];
```

Lista de colunas:

```
(nome_coluna, nome_coluna, ..., nome_coluna)
```

- **INSERT**

- O comando **INSERT** é utilizado para inserir novos registros de dados em uma tabela de um banco de dados.
 - Em outras palavras, inserir uma linha na tabela.
- Sintaxe:

```
INSERT INTO <tabela> [lista de colunas] VALUES  
    <lista de valores>,  
    [lista de valores, ...];
```

Lista de colunas:

```
(nome_coluna, nome_coluna, ..., nome_coluna)
```

A especificação das colunas que terão valores informados é opcional.

Se especificado, as colunas devem ser especificadas entre parênteses e separadas por vírgula.

MySQL

Inserção de dados

- **INSERT**

- O comando **INSERT** é utilizado para inserir novos registros de dados em uma tabela de um banco de dados.
 - Em outras palavras, inserir uma linha na tabela.
- Sintaxe:

```
INSERT INTO <tabela> [lista de colunas] VALUES  
    <lista de valores>,  
    [lista de valores, ...];
```

Lista de colunas:

```
(nome_coluna, nome_coluna, ..., nome_coluna)
```

A especificação dos registros para inserção deverão seguir a mesma ordem da lista das colunas.

Se as colunas não forem especificadas, os registros para inserção devem informar valores para todas as colunas seguindo a ordem em que foram definidas na criação da tabela.

- **INSERT**

- O comando **INSERT** é utilizado para inserir novos registros de dados em uma tabela de um banco de dados.
 - Em outras palavras, inserir uma linha na tabela.
- Sintaxe:

```
INSERT INTO <tabela> [lista de colunas] VALUES  
    <lista de valores>,  
    [lista de valores, ...];
```

Lista de valores:

```
(valor, valor, ..., valor)
```

- **INSERT**

- O comando **INSERT** é utilizado para inserir novos registros de dados em uma tabela de um banco de dados.
 - Em outras palavras, inserir uma linha na tabela.
- Sintaxe:

```
INSERT INTO <tabela> [lista de colunas] VALUES  
  <lista de valores>,  
  [lista de valores, ...];
```

Lista de valores:

```
(valor, valor, ..., valor)
```

Os valores dos campos (colunas) de um registro (linha) devem ser especificados entre parênteses e separados por vírgula.

- **INSERT**

- O comando **INSERT** é utilizado para inserir novos registros de dados em uma tabela de um banco de dados.
 - Em outras palavras, inserir uma linha na tabela.
- Sintaxe:

```
INSERT INTO <tabela> [lista de colunas] VALUES  
  <lista de valores>,  
  [lista de valores, ...];
```

Lista de valores:

```
(valor, valor, ..., valor)
```

Podem ser inseridos mais de um registro (linha) em um único comando **INSERT**. Para isso, basta separar cada registro por vírgula, lembrando que cada registro deve ter seus valores para as colunas entre parênteses.

- **INSERT**

- O comando **INSERT** é utilizado para inserir novos registros de dados em uma tabela de um banco de dados.
 - Em outras palavras, inserir uma linha na tabela.
- Exemplos:

```
INSERT INTO Funcionario (nome, cpf) VALUES
    ("José", "00000000000"),
    ("Maria", "11111111111"),
    ("João", "22222222222");
```

```
INSERT INTO Dependente (id, id_funcionario, nome, cpf) VALUES
    (101, 1, "Carlos", "10101010101"),
    (102, 1, "Ana", "20202020202"),
    (301, 3, "Paulo", "30303030303");
```

- **INSERT**

- O comando **INSERT** é utilizado para inserir novos registros de dados em uma tabela de um banco de dados.
 - Em outras palavras, inserir uma linha na tabela.
- Exemplos:

```
INSERT INTO Funcionario (nome, cpf) VALUES  
    ("José", "00000000000"),  
    ("Maria", "11111111111"),  
    ("João", "22222222222");
```

Observe que a tabela **Funcionario** possui uma coluna **id**, que é a sua chave primária.

Como a coluna **id** foi criada com a especificação **AUTO_INCREMENT**, não precisamos especificar valores para ela.

```
INSERT INTO Dependente (id, id_funcionario, nome, cpf) VALUES  
    (101, 1, "Carlos", "10101010101"),  
    (102, 1, "Ana", "20202020202"),  
    (301, 3, "Paulo", "30303030303");
```

- **INSERT**

- O comando **INSERT** é utilizado para inserir novos registros de dados em uma tabela de um banco de dados.
 - Em outras palavras, inserir uma linha na tabela.
- Exemplos:

```
INSERT INTO Funcionario (nome, cpf) VALUES
    ("José", "00000000000"),
    ("Maria", "11111111111"),
    ("João", "22222222222");
```

A tabela **Dependente** também possui uma coluna **id**, que é a sua chave primária.

Diferente do que ocorre com a tabela **Funcionário**, ela não foi criada com a especificação **AUTO_INCREMENT**. Assim, os valores de **id** devem ser especificados.

```
INSERT INTO Dependente (id, id_funcionario, nome, cpf) VALUES
    (101, 1, "Carlos", "10101010101"),
    (102, 1, "Ana", "20202020202"),
    (301, 3, "Paulo", "30303030303");
```

MySQL

Inserção de dados

- **INSERT**

- O comando **INSERT** é utilizado para inserir novos registros de dados em uma tabela de um banco de dados.

- Em outras palavras, inserir uma linha na tabela.

- Exemplos:

```
INSERT INTO Funcionario (nome, cpf) VALUES
    ("José", "00000000000"),
    ("Maria", "11111111111"),
    ("João", "22222222222");
```

A tabela **Dependente** também possui uma coluna **id**, que é a sua chave primária.

Diferente do que ocorre com a tabela **Funcionário**, ela não foi criada com a especificação **AUTO_INCREMENT**. Assim, os valores de **id** devem ser especificados.

```
INSERT INTO Dependente (id, id_funcionario, nome, cpf) VALUES
    (101, 1, "Carlos", "10101010101"),
    (102, 1, "Ana", "20202020202"),
    (301, 3, "Paulo", "30303030303");
```



- **INSERT**

- O comando **INSERT** é utilizado para inserir novos registros de dados em uma tabela de um banco de dados.
 - Em outras palavras, inserir uma linha na tabela.
- Exemplos:

```
INSERT INTO Funcionario (nome, cpf) VALUES  
  ("José", "00000000000"),  
  ("Maria", "11111111111"),  
  ("João", "22222222222");
```

```
INSERT INTO Dependente (id, id_funcionario, nome, cpf) VALUES  
  (101, 1, "Carlos", "10101010101"),  
  (102, 1, "Ana", "20202020202"),  
  (301, 3, "Paulo", "30303030303");
```

A coluna **id_funcionario** é uma chave estrangeira que indica o funcionário responsável pelo dependente.

- **INSERT**

- O comando **INSERT** é utilizado para inserir novos registros de dados em uma tabela de um banco de dados.
 - Em outras palavras, inserir uma linha na tabela.
- Exemplos:

```
INSERT INTO Funcionario (nome, cpf) VALUES  
  ("José", "00000000000"),  
  ("Maria", "11111111111"),  
  ("João", "22222222222");
```

```
INSERT INTO Dependente (id, id_funcionario, nome, cpf) VALUES  
  (101, 1, "Carlos", "10101010101"),  
  (102, 1, "Ana", "20202020202"),  
  (301, 3, "Paulo", "30303030303");
```

A coluna **id_funcionario** é uma chave estrangeira que indica o funcionário responsável pelo dependente.

MySQL

Inserção de dados

- **INSERT**

- O comando **INSERT** é utilizado para inserir novos registros de dados em uma tabela de um banco de dados.
 - Em outras palavras, inserir uma linha na tabela.
- Exemplo de uso:

```
andre@qwerty: ~  
File Edit View Search Terminal Help  
mysql> INSERT INTO Funcionario (nome, cpf) VALUES  
-> ("José", "00000000000"),  
-> ("Maria", "11111111111"),  
-> ("João", "22222222222");  
Query OK, 3 rows affected (0.01 sec)  
Records: 3 Duplicates: 0 Warnings: 0  
  
mysql>
```

```
andre@qwerty: ~  
File Edit View Search Terminal Help  
mysql> SELECT * FROM Funcionario;  
+-----+-----+-----+  
| id | nome | cpf |  
+-----+-----+-----+  
| 1 | José | 00000000000 |  
| 2 | Maria | 11111111111 |  
| 3 | João | 22222222222 |  
+-----+-----+-----+  
3 rows in set (0.00 sec)  
  
mysql>
```

```
andre@qwerty: ~  
File Edit View Search Terminal Help  
mysql> INSERT INTO Dependente (id, id_funcionario, nome, cpf) VALUES  
-> (101, 1, "Carlos", "10101010101"),  
-> (102, 1, "Ana", "20202020202"),  
-> (301, 3, "Paulo", "30303030303");  
Query OK, 3 rows affected (0.01 sec)  
Records: 3 Duplicates: 0 Warnings: 0  
  
mysql>
```


- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Sintaxe:

```
SELECT [DISTINCT] <lista de dados> FROM <tabela> [, tabela, ...]  
    [WHERE <condição>]  
    [GROUP BY {coluna | expressão}]  
    [HAVING <condição>]  
    [ORDER BY {coluna | expressão} [ASC | DESC]]  
    [LIMIT <num_linhas> [OFFSET <offset>]];
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Sintaxe:

```
SELECT [DISTINCT] <lista de dados> FROM <tabela> [, tabela, ...]  
    [WHERE <condição>]  
    [GROUP BY {coluna | expressão}]  
    [HAVING <condição>]  
    [ORDER BY {coluna | expressão} [ASC | DESC]]  
    [LIMIT <num_linhas> [OFFSET <offset>]];
```

Lista de dados que devem ser retornados pela consulta:

- Determina os dados de interesse (colunas e/ou resultados de expressões) que deverão ser retornados em consulta realizada com o comando **SELECT**.
- Os dados de interesse podem ser especificados um a um, separando-os por vírgulas
 - Por exemplo, os nomes das colunas separados por vírgula.
- Caso o interesse seja em todas as colunas das tabelas envolvidas em uma consulta, pode-se utilizar um asterisco (*****) ao invés de especificar as colunas uma a uma.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Sintaxe:

```
SELECT [DISTINCT] <lista de dados> FROM <tabela> [, tabela, ...]  
    [WHERE <condição>]  
    [GROUP BY {coluna | expressão}]  
    [HAVING <condição>]  
    [ORDER BY {coluna | expressão} [ASC | DESC]]  
    [LIMIT <num_linhas> [OFFSET <offset>]];
```

Modificador **DISTINCT**:

- Por padrão, uma consulta retorna todos os valores que satisfazem as condições (**WHERE** e **HAVING**), inclusive repetições.
- Se o modificador **DISTINCT** for utilizado, as linhas repetidas são omitidas na resposta gerada pela consulta com o comando **SELECT**.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Sintaxe:

```
SELECT [DISTINCT] <lista de dados> FROM <tabela> [, tabela, ...]  
    [WHERE <condição>]  
    [GROUP BY {coluna | expressão}]  
    [HAVING <condição>]  
    [ORDER BY {coluna | expressão} [ASC | DESC]]  
    [LIMIT <num_linhas> [OFFSET <offset>]];
```

Tabelas utilizadas na consulta:

- Toda consulta deve ser realizada a partir de uma ou mais tabelas.
- Se mais de uma tabela for utilizada em uma consulta, estas devem estar separadas por vírgula.
 - Quando mais de uma tabela é especificada, a consulta é realizada sobre o produto cartesiano de seus registros.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Sintaxe:

```
SELECT [DISTINCT] <lista de dados> FROM <tabela> [, tabela, ...]  
    [WHERE <condição>]  
    [GROUP BY {coluna | expressão}]  
    [HAVING <condição>]  
    [ORDER BY {coluna | expressão} [ASC | DESC]]  
    [LIMIT <num_linhas> [OFFSET <offset>]];
```

Tabelas utilizadas na consulta:

Tabela_A

COL_A_1	COL_A_2
x1	y1
x2	y2



Tabela_B

COL_B_1	COL_B_2
w1	z1
w2	z2



Produto cartesiano entre **Tabela1** e **Tabela_B**

COL_A_1	COL_A_2	COL_B_1	COL_B_2
x1	y1	w1	z1
x1	y1	w2	z2
x2	y2	w1	z1
x2	y2	w2	z2

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Sintaxe:

```
SELECT [DISTINCT] <lista de dados> FROM <tabela> [, tabela, ...]  
    [WHERE <condição>]  
    [GROUP BY {coluna | expressão}]  
    [HAVING <condição>]  
    [ORDER BY {coluna | expressão} [ASC | DESC]]  
    [LIMIT <num_linhas> [OFFSET <offset>]];
```

Tabelas utilizadas na consulta:

Tabela_A

COL_A_1	COL_A_2
x1	y1
x2	y2



Tabela_B

COL_B_1	COL_B_2
w1	z1
w2	z2



Produto cartesiano entre **Tabela1** e **Tabela_B**

COL_A_1	COL_A_2	COL_B_1	COL_B_2
x1	y1	w1	z1
x1	y1	w2	z2
x2	y2	w1	z1
x2	y2	w2	z2

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Sintaxe:

```
SELECT [DISTINCT] <lista de dados> FROM <tabela> [, tabela, ...]  
    [WHERE <condição>]  
    [GROUP BY {coluna | expressão}]  
    [HAVING <condição>]  
    [ORDER BY {coluna | expressão} [ASC | DESC]]  
    [LIMIT <num_linhas> [OFFSET <offset>]];
```

Tabelas utilizadas na consulta:

Tabela_A

COL_A_1	COL_A_2
x1	y1
x2	y2



Tabela_B

COL_B_1	COL_B_2
w1	z1
w2	z2



Produto cartesiano entre **Tabela1** e **Tabela_B**

COL_A_1	COL_A_2	COL_B_1	COL_B_2
x1	y1	w1	z1
x1	y1	w2	z2
x2	y2	w1	z1
x2	y2	w2	z2

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Sintaxe:

```
SELECT [DISTINCT] <lista de dados> FROM <tabela> [, tabela, ...]  
    [WHERE <condição>]  
    [GROUP BY {coluna | expressão}]  
    [HAVING <condição>]  
    [ORDER BY {coluna | expressão} [ASC | DESC]]  
    [LIMIT <num_linhas> [OFFSET <offset>]];
```

Tabelas utilizadas na consulta:

Tabela_A

COL_A_1	COL_A_2
x1	y1
x2	y2



Tabela_B

COL_B_1	COL_B_2
w1	z1
w2	z2



Produto cartesiano entre **Tabela1** e **Tabela_B**

COL_A_1	COL_A_2	COL_B_1	COL_B_2
x1	y1	w1	z1
x1	y1	w2	z2
x2	y2	w1	z1
x2	y2	w2	z2

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Sintaxe:

```
SELECT [DISTINCT] <lista de dados> FROM <tabela> [, tabela, ...]  
    [WHERE <condição>]  
    [GROUP BY {coluna | expressão}]  
    [HAVING <condição>]  
    [ORDER BY {coluna | expressão} [ASC | DESC]]  
    [LIMIT <num_linhas> [OFFSET <offset>]];
```

Tabelas utilizadas na consulta:

Tabela_A

COL_A_1	COL_A_2
x1	y1
x2	y2



Tabela_B

COL_B_1	COL_B_2
w1	z1
w2	z2



Produto cartesiano entre **Tabela1** e **Tabela_B**

COL_A_1	COL_A_2	COL_B_1	COL_B_2
x1	y1	w1	z1
x1	y1	w2	z2
x2	y2	w1	z1
x2	y2	w2	z2

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Sintaxe:

```
SELECT [DISTINCT] <lista de dados> FROM <tabela> [, tabela, ...]  
[WHERE <condição>]  
[GROUP BY {coluna | expressão}]  
[HAVING <condição>]  
[ORDER BY {coluna | expressão} [ASC | DESC]]  
[LIMIT <num_linhas> [OFFSET <offset>]];
```

Cláusula **WHERE**:

- A cláusula WHERE, se especificada, indica a condição (ou condições) que devem ser satisfeitas pelos registros que serão retornados pela consulta.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Sintaxe:

```
SELECT [DISTINCT] <lista de dados> FROM <tabela> [, tabela, ...]  
    [WHERE <condição>]  
    [GROUP BY {coluna | expressão}]  
    [HAVING <condição>]  
    [ORDER BY {coluna | expressão} [ASC | DESC]]  
    [LIMIT <num_linhas> [OFFSET <offset>]];
```

Cláusula **GROUP BY**:

- A cláusula GROUP BY é utilizada para realizar o agrupamento de dados.
 - O agrupamento de dados é utilizado quando funções de agregação são aplicadas sobre colunas:
 - **AVG** : cálculo da média.
 - **SUM** : somatório.
 - **MAX** : maior valor.
 - **MIN** : menor valor.
 - **COUNT** : conta o número de linhas.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Sintaxe:

```
SELECT [DISTINCT] <lista de dados> FROM <tabela> [, tabela, ...]  
    [WHERE <condição>]  
    [GROUP BY {coluna | expressão}]  
    [HAVING <condição>]  
    [ORDER BY {coluna | expressão} [ASC | DESC]]  
    [LIMIT <num_linhas> [OFFSET <offset>]];
```

Cláusula **HAVING**:

- A cláusula HAVING foi introduzida ao SQL pois a cláusula WHERE não suporta o uso de funções de agregação ao filtrar os resultados.
- A cláusula HAVING não deve ser usada com condições suportadas pela cláusula WHERE.
- O padrão SQL especifica que a cláusula HAVING deve ser utilizada apenas em colunas presentes utilizadas com a cláusula WHERE ou com colunas utilizadas em funções de agregação.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Sintaxe:

```
SELECT [DISTINCT] <lista de dados> FROM <tabela> [, tabela, ...]  
    [WHERE <condição>]  
    [GROUP BY {coluna | expressão}]  
    [HAVING <condição>]  
    [ORDER BY {coluna | expressão} [ASC | DESC]]  
    [LIMIT <num_linhas> [OFFSET <offset>]];
```

Cláusula **ORDER BY**:

- Ordena os resultados retornados pela consulta.
- A ordenação pode ser realizada em relação a uma ou mais colunas/expressões.
 - Se especificada mais de uma coluna/expressão, elas devem ser separadas por vírgula.
- Pode ser especificado, para cada coluna/expressão, se a ordenação será realizada em ordem ascendente (ASC, que é utilizado por padrão, caso não seja informada) ou descendente (DESC).

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Sintaxe:

```
SELECT [DISTINCT] <lista de dados> FROM <tabela> [, tabela, ...]  
    [WHERE <condição>]  
    [GROUP BY {coluna | expressão}]  
    [HAVING <condição>]  
    [ORDER BY {coluna | expressão} [ASC | DESC]]  
    [LIMIT <num_linhas> [OFFSET <offset>]];
```

Cláusula **LIMIT**:

- A cláusula **LIMIT** determina o número máximo de registros (linhas) que deverão ser retornados pela consulta.
 - Por padrão, são retornados os primeiros registros.
- O modificador **OFFSET** é utilizado para que os **offset** primeiros resultados sejam ignorados. Exemplo onde são retornados os registros de 6 a 15:
LIMIT 10 OFFSET 5

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Sintaxe:

```
SELECT [DISTINCT] <lista de dados> FROM <tabela> [, tabela, ...]  
    [WHERE <condição>]  
    [GROUP BY {coluna | expressão}]  
    [HAVING <condição>]  
    [ORDER BY {coluna | expressão} [ASC | DESC]]  
    [LIMIT <num_linhas> [OFFSET <offset>]];
```

As cláusulas e modificadores apresentados, quando utilizados, devem, em geral, ser especificados nesta ordem.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo:
 - Criação do banco de dados

```
CREATE DATABASE empresa;  
USE empresa;
```


- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Criação das tabelas:

```
CREATE TABLE Departamento (  
    id INTEGER PRIMARY KEY AUTO_INCREMENT,  
    nome VARCHAR(30) NOT NULL  
);
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Criação das tabelas:

```
CREATE TABLE Funcionario (  
    id INTEGER PRIMARY KEY AUTO_INCREMENT,  
    nome VARCHAR(50) NOT NULL,  
    sobrenome VARCHAR(50) NOT NULL,  
    salario REAL NOT NULL,  
    id_dept INTEGER NOT NULL,  
    CONSTRAINT FOREIGN KEY (id_dept)  
        REFERENCES Departamento (id)  
);
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Criação das tabelas:

```
CREATE TABLE Dependente (  
    id INTEGER AUTO_INCREMENT,  
    id_fun INTEGER NOT NULL,  
    nome VARCHAR(50) NOT NULL,  
    sobrenome VARCHAR(50) NOT NULL,  
    CONSTRAINT PRIMARY KEY (id, id_fun),  
    CONSTRAINT FOREIGN KEY (id_fun)  
        REFERENCES Funcionario (id)  
);
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Inserção de dados:

```
INSERT INTO Departamento (nome) VALUES
  ("Administrativo"),
  ("Financeiro"),
  ("Recursos Humanos"),
  ("Comercial"),
  ("Pesquisa"),
  ("Tecnologia da Informação");
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Inserção de dados:

```
INSERT INTO Funcionario (nome, sobrenome, id_dept, salario) VALUES
  ("Adolfo",      "Feijó",      1, 7000.0),
  ("Adriana",     "Albuquerque", 1, 7000.0),
  ("Adélio",      "Saldaña",    2, 5000.0),
  ("Catarina",    "Javier",     2, 4000.0),
  ("Cosperranho", "Lira",       3, 3500.0),
  ("Doroteia",    "Almeida",    4, 2000.0),
  ("Eládio",      "Parreira",   4, 2000.0),
  ("Evandro",     "Beiriz",     5, 6500.0),
  ("Evangelista", "Hollanda",   5, 7000.0),
  ("Gabriela",    "Brasil",     5, 4200.0),
  ("Heitor",      "Seixas",     5, 6500.0),
  ("Inês",        "Belém",      6, 4000.0),
  ("Lourenço",    "Bicalho",    6, 2000.0),
  ("Marlene",     "Vidigal",    6, 2000.0);
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Inserção de dados:

```
INSERT INTO Dependente (nome, sobrenome, id_fun) VALUES
  ("Milena",      "Albuquerque",  2),
  ("Osvaldo",     "Albuquerque",  2),
  ("Sidónio",     "Almeida",      6),
  ("Sérgio",      "Bicalho",     13),
  ("Teodorico",   "Bicalho",     13),
  ("Virgínia",    "Belém",       12);
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar todos os dados dos funcionários.

```
SELECT * FROM Funcionario;
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar apenas o nome e o salário dos funcionários.

```
SELECT nome, salario FROM Funcionario;
```


- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar nome, sobrenome e salário dos funcionários, combinando o nome e o sobrenome em uma única coluna.

```
SELECT CONCAT(nome, " ", sobrenome), salario FROM Funcionario;
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar nome, sobrenome e salário dos funcionários, combinando o nome e o sobrenome em uma única coluna.

```
SELECT CONCAT(nome, " ", sobrenome), salario FROM Funcionario;
```

A função **CONCAT** pode ser utilizada para concatenar dados do tipo texto.

O resultado será apresentado em uma coluna com o nome igual o da expressão.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar nome, sobrenome e salário dos funcionários, combinando o nome e o sobrenome em uma única coluna.

```
SELECT CONCAT(nome, " ", sobrenome) AS "nome_completo", salario FROM Funcionario;
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar nome, sobrenome e salário dos funcionários, combinando o nome e o sobrenome em uma única coluna.

```
SELECT CONCAT(nome, " ", sobrenome) AS "nome_completo", salario FROM Funcionario;
```

Nesse exemplo, utilizamos o comando **AS** para alterar o nome da coluna.

O comando **AS** também pode ser utilizado com colunas além de expressões.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados dos funcionários que recebem um salário maior ou igual a 5000.

```
SELECT * FROM Funcionario  
WHERE salario >= 5000;
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados dos funcionários que recebem um salário maior ou igual a 5000 e trabalham no Departamento de Pesquisa (id 5).

```
SELECT * FROM Funcionario  
WHERE salario >= 5000 AND id_dept = 5;
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados dos funcionários que recebem um salário maior ou igual a 5000 e trabalham no Departamento de Pesquisa (id 5).

```
SELECT * FROM Funcionario  
WHERE salario >= 5000 AND id_dept = 5;
```

Podemos utilizar operador lógico **AND** quando desejamos que duas condições sejam satisfeitas.

Podemos também utilizar o operador lógico **OR** quando desejamos que pelo menos uma condição seja satisfeita.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados dos funcionários que trabalham no Departamento de Tecnologia da Informação (id 6) ou no Departamento de Pesquisa (id 5).

```
SELECT * FROM Funcionario  
WHERE id_dept = 6 OR id_dept = 5;
```


- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados dos funcionários e seus respectivos departamentos.

```
SELECT * FROM Funcionario, Departamento;
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados dos funcionários e seus respectivos departamentos.

```
SELECT * FROM Funcionario, Departamento;
```

O resultado será o produto cartesiano.

Precisamos filtrar o resultado para que cada funcionário apareça apenas com o seu departamento.

id	nome	sobrenome	salario	id_dept	id	nome
1	Adolfo	Feijó	7000	1	1	Administrativo
1	Adolfo	Feijó	7000	1	2	Financeiro
1	Adolfo	Feijó	7000	1	3	Recursos Humanos
1	Adolfo	Feijó	7000	1	4	Comercial
1	Adolfo	Feijó	7000	1	5	Pesquisa
1	Adolfo	Feijó	7000	1	6	Tecnologia da Informação
2	Adriana	Albuquerque	7000	1	1	Administrativo
2	Adriana	Albuquerque	7000	1	2	Financeiro

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados dos funcionários e seus respectivos departamentos.

```
SELECT * FROM Funcionario, Departamento  
WHERE Funcionario.id_dept = Departamento.id;
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados dos funcionários e seus respectivos departamentos.

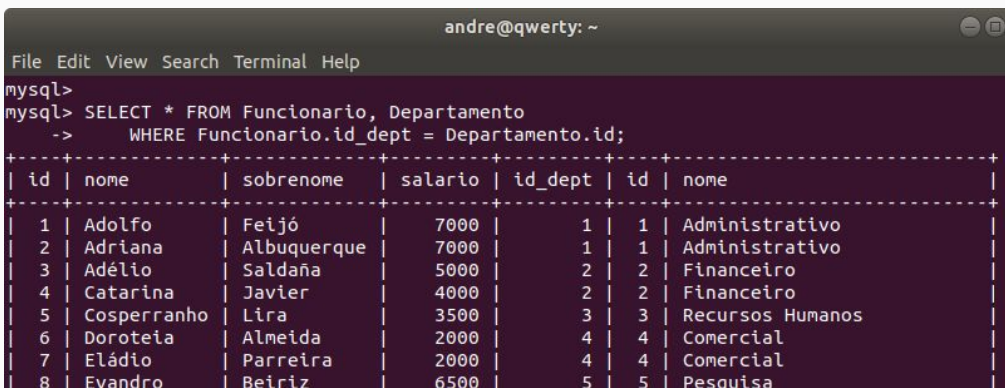
```
SELECT * FROM Funcionario, Departamento  
WHERE Funcionario.id_dept = Departamento.id;
```

Devemos retornar apenas as linhas em que a coluna **id_dept** de **Funcionario** for igual à coluna **id** de **Departamento**.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados dos funcionários e seus respectivos departamentos.

```
SELECT * FROM Funcionario, Departamento  
WHERE Funcionario.id_dept = Departamento.id;
```



A terminal window titled 'andre@qwerty: ~' shows the execution of a MySQL query. The prompt is 'mysql>'. The query entered is 'SELECT * FROM Funcionario, Departamento WHERE Funcionario.id_dept = Departamento.id;'. The output is a table with 8 rows and 7 columns. The columns are 'id', 'nome', 'sobrenome', 'salario', 'id_dept', 'id', and 'nome'. The rows represent employees and their departments.

id	nome	sobrenome	salario	id_dept	id	nome
1	Adolfo	Feijó	7000	1	1	Administrativo
2	Adriana	Albuquerque	7000	1	1	Administrativo
3	Adélio	Saldanha	5000	2	2	Financeiro
4	Catarina	Javier	4000	2	2	Financeiro
5	Cosperranho	Lira	3500	3	3	Recursos Humanos
6	Doroteta	Almeida	2000	4	4	Comercial
7	Eládio	Parreira	2000	4	4	Comercial
8	Evandro	Beiriz	6500	5	5	Pesquisa

Devemos retornar apenas as linhas em que a coluna **id_dept** de **Funcionario** for igual à coluna **id** de **Departamento**.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados dos funcionários e seus respectivos departamentos.

```
SELECT * FROM Funcionario, Departamento  
WHERE Funcionario.id_dept = Departamento.id;
```

Quando utilizamos múltiplas tabelas em uma consulta, é comum utilizarmos o nome da tabela junto ao nome da coluna.

Quando as tabelas possuem colunas com mesmo nome, a especificação da tabela junto ao nome da coluna é necessário.

Porém, nomes grandes de tabelas podem deixar o código da consulta verboso.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados dos funcionários e seus respectivos departamentos.

```
SELECT * FROM Funcionario AS f, Departamento AS d
WHERE f.id_dept = d.id;
```

Uma alternativa para facilitar a escrita, é atribuir nomes **apelidos** às tabelas na consulta.

Isso é feito com o uso do **AS**.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados de todos os funcionários que o nome começa com a letra **A**.

```
SELECT * FROM Funcionario  
WHERE nome LIKE "A%";
```


- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados de todos os funcionários que o nome começa com a letra **A**.

```
SELECT * FROM Funcionario  
WHERE nome LIKE "A%";
```

O operador **LIKE** é utilizado para casamento de padrões em cadeias de caracteres.

Se utilizado o operador de igualdade (=), a comparação é feita caractere por caractere.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados de todos os funcionários que o nome começa com a letra **A**.

```
SELECT * FROM Funcionario  
WHERE nome LIKE "A%";
```

Quando utilizado em conjunto com o operador **LIKE**, o caractere % (porcentagem) significa qualquer sequência de caracteres (inclusive vazia).

O caractere _ (*underline*) significa um caractere qualquer.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pela soma dos salários de todos os funcionários da empresa.

```
SELECT SUM(salario) AS "soma_salarios" FROM Funcionario;
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pela soma dos salários de todos os funcionários por departamento.

```
SELECT Departamento.nome AS "departamento", SUM(salario) AS "soma_salarios"  
FROM Funcionario, Departamento  
WHERE Funcionario.id_dept = Departamento.id  
GROUP BY departamento;
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pela soma dos salários de todos os funcionários por departamento e que a soma dos salários seja maior que 10000.

```
SELECT Departamento.nome AS "departamento", SUM(salario) AS "soma_salarios"
FROM Funcionario, Departamento
WHERE Funcionario.id_dept = Departamento.id
GROUP BY departamento
HAVING SUM(salario) > 10000;
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pela soma dos salários de todos os funcionários por departamento e que a soma dos salários seja maior que 10000.

```
SELECT Departamento.nome AS "departamento", SUM(salario) AS "soma_salarios"  
FROM Funcionario, Departamento  
WHERE Funcionario.id_dept = Departamento.id  
GROUP BY departamento  
HAVING SUM(salario) > 10000;
```

Lembre-se que quando queremos utilizar uma função de agregação para filtrar a resultado, utilizamos a cláusula **HAVING**.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos funcionários da empresa, ordenando-os pelo nome em ordem decrescente.

```
SELECT * FROM Funcionario  
ORDER BY nome DESC;
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos funcionários da empresa, ordenando-os pelo nome em ordem decrescente, limitando os resultados a apenas 3 linhas.

```
SELECT * FROM Funcionario  
ORDER BY nome DESC  
LIMIT 3;
```


- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos funcionários da empresa, ordenando-os pelo nome em ordem decrescente, limitando os resultados a apenas 3 linhas iniciando da quinta linha.

```
SELECT * FROM Funcionario  
ORDER BY nome DESC  
LIMIT 3 OFFSET 4;
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados dos funcionários que possuem algum dependente.

```
SELECT Funcionario.* FROM Funcionario, Dependente  
WHERE Funcionario.id = Dependente.id_fun;
```

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados dos funcionários que possuem algum dependente.

```
SELECT Funcionario.* FROM Funcionario, Dependente
WHERE Funcionario.id = Dependente.id_fun;
```

```
andre@qwerty: ~
File Edit View Search Terminal Help
mysql> SELECT Funcionario.* FROM Funcionario, Dependente
WHERE Funcionario.id = Dependente.id_fun;
+-----+-----+-----+-----+-----+
| id | nome      | sobrenome | salario | id_dept |
+-----+-----+-----+-----+-----+
| 2  | Adriana   | Albuquerque | 7000    | 1        |
| 2  | Adriana   | Albuquerque | 7000    | 1        |
| 6  | Doroteia  | Almeida    | 3000    | 4        |
| 12 | Inês      | Belém      | 4000    | 6        |
| 13 | Lourenço  | Bicalho    | 3000    | 6        |
| 13 | Lourenço  | Bicalho    | 3000    | 6        |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

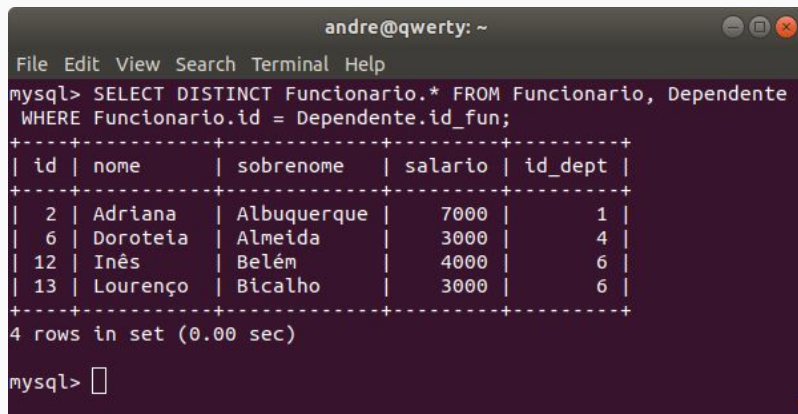
mysql>
```

O resultado apresentará valores repetidos. Um para cada combinação de um funcionário com seu respectivo Dependente.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados dos funcionários que possuem algum dependente.

```
SELECT DISTINCT Funcionario.* FROM Funcionario, Dependente  
WHERE Funcionario.id = Dependente.id_fun;
```



A terminal window titled 'andre@qwerty: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is 'mysql>'. The user enters the query: 'SELECT DISTINCT Funcionario.* FROM Funcionario, Dependente WHERE Funcionario.id = Dependente.id_fun;'. The output is a table with 5 columns: id, nome, sobrenome, salario, id_dept. It contains 4 rows of data. Below the table, it says '4 rows in set (0.00 sec)'. The prompt 'mysql>' is shown again at the bottom.

```
mysql> SELECT DISTINCT Funcionario.* FROM Funcionario, Dependente  
WHERE Funcionario.id = Dependente.id_fun;  
+-----+-----+-----+-----+-----+  
| id | nome      | sobrenome | salario | id_dept |  
+-----+-----+-----+-----+-----+  
| 2  | Adriana   | Albuquerque | 7000    | 1        |  
| 6  | Doroteia  | Almeida    | 3000    | 4        |  
| 12 | Inês      | Belém      | 4000    | 6        |  
| 13 | Lourenço  | Bicalho    | 3000    | 6        |  
+-----+-----+-----+-----+-----+  
4 rows in set (0.00 sec)  
  
mysql> 
```

A repetição pode ser resolvida com o uso da opção **DISTINCT**.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados dos funcionários que possuem algum dependente.

```
SELECT * FROM Funcionario
WHERE id IN (SELECT f.id FROM Funcionario AS f, Dependente AS d
            WHERE f.id = d.id_fun);
```

```
andre@qwerty: ~
File Edit View Search Terminal Help
mysql> SELECT * FROM Funcionario WHERE id IN (SELECT f.id FROM Funcionario AS f, Dependente AS d WHERE f.id = d.id_fun);
+-----+-----+-----+-----+-----+
| id | nome   | sobrenome | salario | id_dept |
+-----+-----+-----+-----+-----+
| 2  | Adriana | Albuquerque | 7000    | 1       |
| 6  | Doroteia | Almeida   | 3000    | 4       |
| 12 | Inês    | Belém     | 4000    | 6       |
| 13 | Lourenço | Bicalho   | 3000    | 6       |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

Outra forma de realizar essa consulta é utilizando **subqueries**.

- **SELECT**

- O comando **SELECT** é utilizado realizar consultas aos dados armazenados.
- Exemplo (continuação):
 - Consulta:
 - Buscar pelos dados dos funcionários que possuem não têm dependentes.

```
SELECT * FROM Funcionario
WHERE id NOT IN (SELECT f.id FROM Funcionario AS f, Dependente AS d
                WHERE f.id = d.id_fun);
```