

# Introdução a Linguagem SQL



Sicilia Giacomazza

Introdução à Banco de Dados com MySQL

2022

# Sumário

Sumário .....	2
Introdução à Linguagem SQL.....	4
DML - Linguagem de Manipulação de Dados .....	4
DDL - Linguagem de Definição de Dados .....	4
DQL - Linguagem de Consulta de Dados.....	4
DQL: Comando de seleção .....	5
DQL: Buscas e cláusulas .....	5
DML: Comando de inserção.....	6
DML: Comando de atualização .....	7
DML: DROP versus DELETE .....	7
O que é MySQL?.....	8
Criando um Schema.....	8
Definindo ESCOLA_CURSO como Padrão .....	9
Criando Tabelas .....	9
Alterando Tabelas .....	10
Backup de Tabelas.....	11
Apagando Tabelas .....	12
Limpando a Tabela.....	12
Alterando dados em uma Tabela .....	13
Deletando dados em uma Tabela .....	13
Criando outras Tabelas .....	14
Inserindo dados nas Tabelas .....	15
Realizando consultas nas Tabelas.....	18
Operadores Lógicos e Racionais .....	21

Cardinalidade – Relacionamentos.....	23
Criando Chaves Estrangeiras.....	25
Criando outras Tabelas com suas FK .....	27
Joins .....	29
Estudo de Caso: Base de Dados de Filmes .....	30
Criando a Database Filmes .....	32
Criando a Tabela Filmes .....	33
Importando o arquivo .csv Filmes.....	34
Análise da Base de Dados Filmes.....	35

# Introdução à Linguagem SQL

**SQL (*Structured Query Language*)** quer dizer Linguagem de Consulta Estruturada. Permite a manipulação de tabelas do banco de dados. Ela é a linguagem de busca de informações em bancos de dados relacionais. A linguagem SQL é dividida em:

## DML - Linguagem de Manipulação de Dados

Permite manipulação de dados, como exclusão, inclusão e alterações. Exemplos de comandos:

- INSERT (permite adicionar dados)
- UPDATE (permite atualizar dados)
- DELETE (permite apagar dados)

## DDL - Linguagem de Definição de Dados

Permite a criação e alteração de dados. Exemplos de comandos:

- CREATE TABLE (cria tabelas)
- ALTER TABLE (altera tabelas)
- DROP TABLE (apaga tabelas).

## DQL - Linguagem de Consulta de Dados

Permite a realização de buscas nas tabelas dos bancos de dados. Exemplo de comando:

- SELECT (comando mais importante usado para realizar buscas)

## DQL: Comando de seleção

### SELECT: SELECIONAR

Na linguagem SQL, podemos utilizar o comando SELECT para ler dados de tabelas. SELECT é um comando do tipo DQL (*Data Query Language* - Linguagem de Consulta de dados). Veja os exemplos a seguir:

```
SELECT * FROM MINHA_TABELA
```

(traduzido literalmente seria "SELECIONAR \* DA MINHA\_TABELA", onde asterisco representa "TUDO")

Você poderia especificar colunas que deseja fazer a seleção, alterando o \*. Por exemplo:

```
SELECT NOME FROM ALUNOS
```

("SELECIONAR (coluna) NOME DA (tabela) ALUNOS")

### DQL: Buscas e cláusulas

Buscas podem ser melhoradas com cláusulas:

```
SELECT * FROM TABELA
```

Realiza uma busca por todos os dados `*` em uma tabela chamada TABELA. A cláusula `FROM` indica a tabela.

Ainda há outras cláusulas:

- WHERE: indica as condições
- GROUP BY: realiza agrupamentos
- ORDER BY: ordena os dados

Ainda podemos combinar buscas com operadores lógicos.

- AND: avalia se duas condições são verdadeiras
- OR: avalia se uma condição é verdadeira
- NOT: negação

Operadores relacionais permitem fazer comparações nas consultas:

- < Menor
- > Maior
- <= Menor ou igual
- >= Maior ou igual
- = Igual
- <> Diferente

## DML: Comando de inserção

### INSERT: INSERIR

Na linguagem SQL, podemos utilizar o comando INSERT para inserir dados em uma tabela. INSERT é um comando do tipo **DML** (*Data Manipulation Language* - Linguagem de Manipulação de Dados). Veja os exemplos a seguir:

```
INSERT INTO MINHA_TABELA (CAMPOS) VALUES("VALORES")
```

(traduzido literalmente seria "INSERIR DENTRO DA MINHA TABELA (NOME DAS COLUNAS QUE QUER FAZER INSERÇÃO) OS VALORES ('COLOCAR VALORES ENTRE ASPAS OU APÓSTROFOS)').

\* Observe que números não precisam de aspas.

## DML: Comando de atualização

### UPDATE: ATUALIZAR

Na linguagem SQL, podemos utilizar o comando UPDATE para atualizar dados em uma tabela. UPDATE é um comando do tipo **DML** (*Data Manipulation Language* - Linguagem de Manipulação de Dados). Veja os exemplos a seguir:

```
UPDATE MINHA_TABELA SET CAMPO="NOVO VALOR" WHERE ID =1
```

(traduzido literalmente seria "ATUALIZAR MINHA TABELA DEFINA O CAMPO = "NOVO VALOR" ONDE ID=1)

\* Neste exemplo, o ID representa qual linha será atualizada. No caso estamos atualizando a coluna chamada CAMPO na primeira linha.

### DML: DROP versus DELETE

**DROP:** do inglês derrubar, soltar, jogar

**DELETE:** do inglês apagar, deletar

Na linguagem SQL, DROP é um comando do tipo DDL, ou seja, comando de definição de dados; enquanto, DELETE é um comando do tipo DML, ou seja, manipulação de dados.

- Use DROP para excluir tabelas e bases de dados;
- Use DELETE para deletar dados em tabelas.

## O que é MySQL?

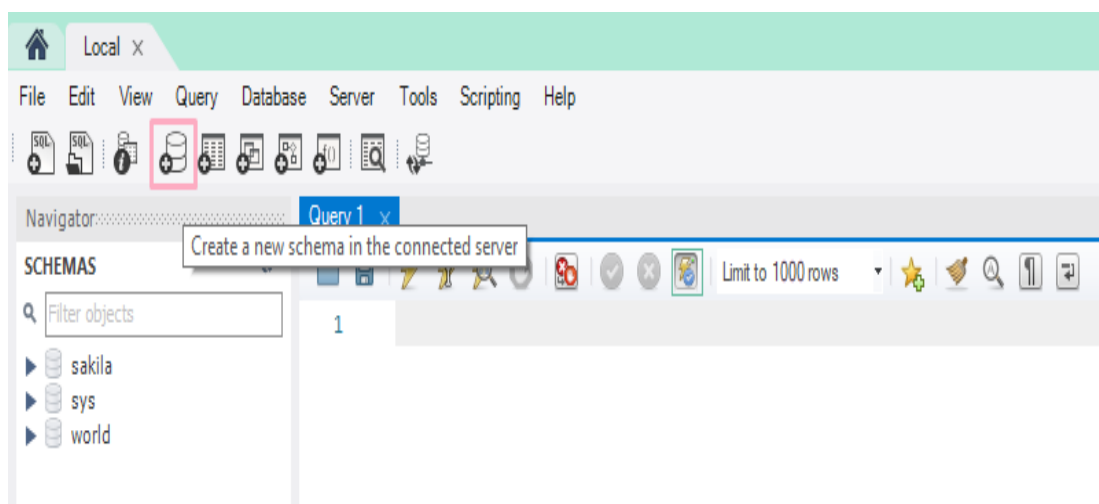


**MySQL** é um sistema de gerenciamento de bancos de dados de grande popularidade.

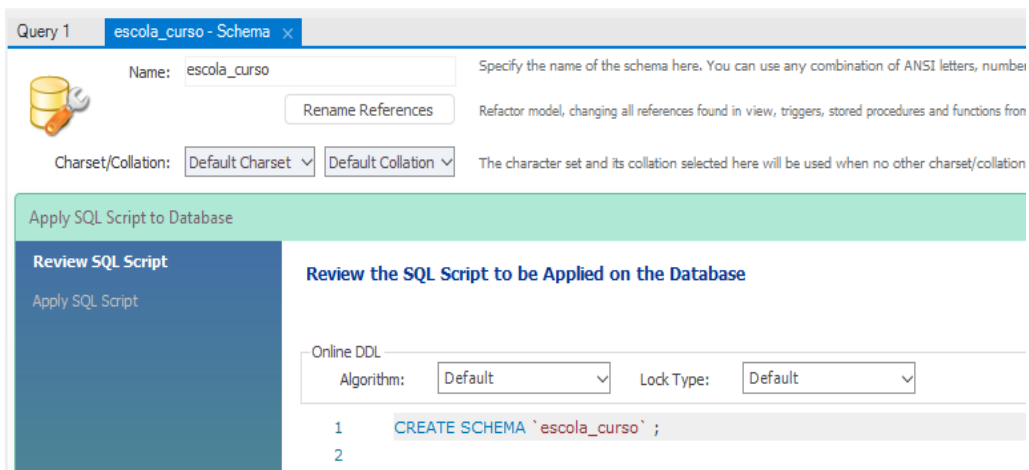
Dentre suas principais características destacam-se:

- **Interoperabilidade:** roda em diversos sistemas operacionais;
- **Compatibilidade com diversas linguagens de programação:** possui módulos que permitem a integração com linguagens como PHP, Python, Java, Perl, C/C++, ASP, dentre outras;
- **Comunidade ativa:** possui uma vasta comunidade de usuários, o que facilita o suporte;
- **É gratuito!**

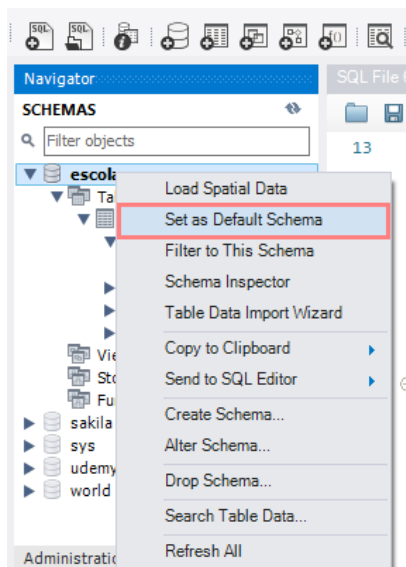
## Criando um Schema



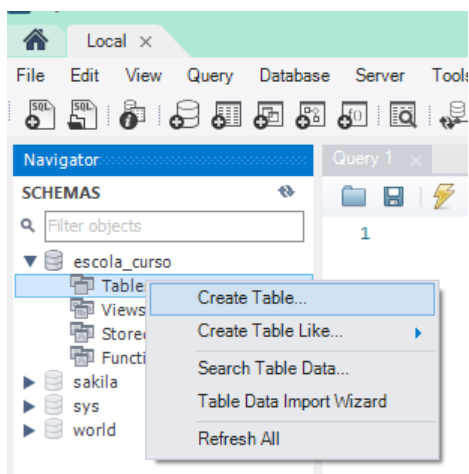




## Definindo ESCOLA\_CURSO como Padrão



## Criando Tabelas



PK (*Primary Key*)

NN (*Not Null*)

AI (*Auto Incremental*)

```
CREATE TABLE `alunos` (
```

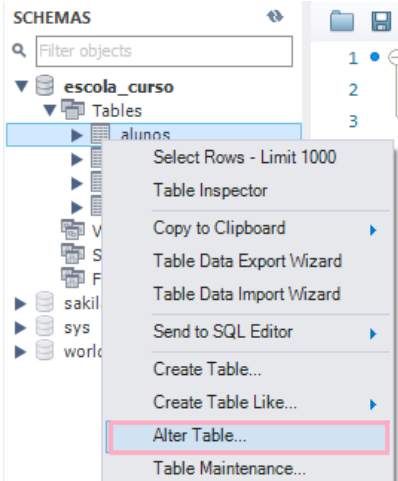
```
`id_aluno` int NOT NULL AUTO_INCREMENT,
```

```
PRIMARY KEY (`id_aluno`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
```

```
COLLATE=utf8mb4_0900_ai_ci;
```

## Alterando Tabelas



The screenshot shows the MySQL Workbench interface. In the 'SCHEMAS' panel on the left, the 'escola\_curso' database is selected, and the 'alunos' table is highlighted. A right-click context menu is open over the 'alunos' table, with the 'Alter Table...' option highlighted in red. Below the menu, the 'alunos - Table' tab is active in the main workspace. The 'Table Name' is 'alunos', and the 'Charset/Collation' is 'utf8mb4'. The 'Comments' field is empty. Below this, a table lists the columns of the 'alunos' table with their data types and constraints.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
id_aluno	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
nome	VARCHAR(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
data_nascimento	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
endereço	VARCHAR(255)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
cidade	VARCHAR(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
estado	VARCHAR(2)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
cpf	VARCHAR(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

```
ALTER TABLE `escola_curso`.`alunos`
```

```
ADD COLUMN `nome` VARCHAR(100) NOT NULL AFTER `id_aluno`,
```

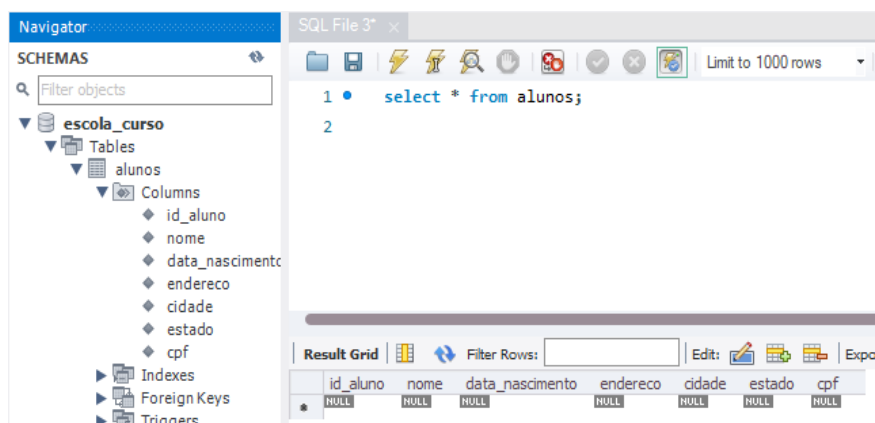
```
ADD COLUMN `data_nascimento` DATE NOT NULL AFTER `nome`,
```

```
ADD COLUMN `endereco` VARCHAR(255) NOT NULL AFTER  
`data_nascimento`,
```

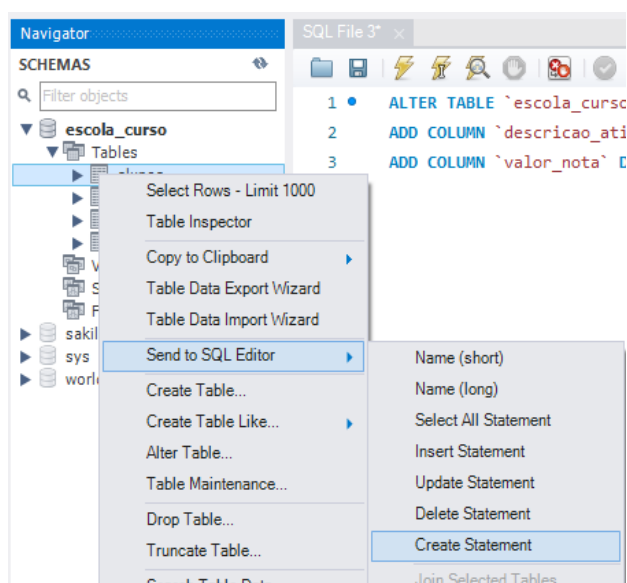
```
ADD COLUMN `cidade` VARCHAR(100) NOT NULL AFTER `endereco`,
```

```
ADD COLUMN `estado` VARCHAR(2) NOT NULL AFTER `cidade`,
```

```
ADD COLUMN `cpf` VARCHAR(11) NOT NULL AFTER `estado`;
```

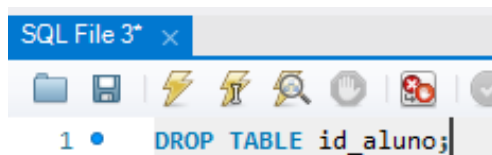


## Backup de Tabelas



```
CREATE TABLE `alunos` (  
  
  `id_aluno` int NOT NULL AUTO_INCREMENT,  
  
  `nome` varchar(100) NOT NULL,  
  
  `data_nascimento` date NOT NULL,  
  
  `endereco` varchar(255) NOT NULL,  
  
  `cidade` varchar(100) NOT NULL,  
  
  `estado` varchar(2) NOT NULL,  
  
  `cpf` varchar(11) NOT NULL,  
  
  PRIMARY KEY (`id_aluno`)  
  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4  
COLLATE=utf8mb4_0900_ai_ci;
```

## Apagando Tabelas



```
DROP TABLE id_aluno;
```

## Limpendo a Tabela

```
TRUNCATE TABLE alunos;
```

## Alterando dados em uma Tabela

UPDATE alunos

SET nome = 'Pedro Magalhães Martins'

WHERE id\_aluno = 1;

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' panel displays the structure of the 'escola\_curso' database, including the 'alunos' table with columns: id\_aluno, nome, data\_nascimento, endereco, cidade, estado, and cpf. On the right, the SQL editor shows the query: `SELECT * FROM escola_curso.alunos;`. Below the editor, the 'Result Grid' displays the query results in a table format.

id_aluno	nome	data_nascimento	endereco	cidade	estado	cpf
1	Pedro Magalhães Martins	1987-07-17	Av. Ant. Carlos, 6673	BELO HORIZONTE	MG	12345678911

## Deletando dados em uma Tabela

DELETE FROM alunos

WHERE id\_aluno = 1;

The screenshot shows a database management tool interface. The 'alunos' table is selected in the left panel. The SQL editor shows the query: `SELECT id_aluno FROM `escola_curso`.`alunos`;`. Below the editor, the 'Result Grid' displays the query results, showing a list of id\_aluno values: 2, 3, 4, 5, and NULL.

id_aluno
2
3
4
5
NULL

## Criando outras Tabelas

Primeiramente, delete a Tabela Alunos e crie uma nova com a query a seguir:

### # CRIANDO TABELA ALUNOS

```
CREATE TABLE `alunos` (  
  `id_aluno` int NOT NULL AUTO_INCREMENT,  
  `nome` varchar(100) NOT NULL,  
  `data_nascimento` date NOT NULL,  
  `endereco` varchar(255) NOT NULL,  
  `cidade` varchar(100) NOT NULL,  
  `estado` varchar(2) NOT NULL,  
  `cpf` varchar(11) NOT NULL,  
  PRIMARY KEY (`id_aluno`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

### # CRIANDO TABELA CURSOS

```
CREATE TABLE `cursos` (  
  `id_curso` int NOT NULL AUTO_INCREMENT,  
  `nome` varchar(100) NOT NULL,  
  PRIMARY KEY (`id_curso`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

### # CRIANDO TABELA ALUNOS\_CURSOS

```
CREATE TABLE `alunos_cursos` (  

```

```
`id_aluno_curso` int NOT NULL AUTO_INCREMENT,
```

```
`id_aluno` int NOT NULL,
```

```
`id_curso` int NOT NULL,
```

```
PRIMARY KEY (`id_aluno_curso`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

## # CRIANDO TABELA NOTAS

```
CREATE TABLE `notas` (
```

```
`id_nota` int NOT NULL AUTO_INCREMENT,
```

```
`id_aluno_curso` int NOT NULL,
```

```
`descricao_atividade` varchar(100) NOT NULL,
```

```
`vlr_nota` decimal(5,2) NOT NULL,
```

```
PRIMARY KEY (`id_nota`)
```

```
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

## Inserindo dados nas Tabelas

### # POVOANDO TABELAS ALUNOS

```
INSERT INTO alunos VALUES
```

```
(DEFAULT,'Pedro Martins', '1987-07-17', 'Av. Ant. Carlos, 6673', 'BELO
```

```
HORIZONTE', 'MG', '12345678911'),
```

```
(DEFAULT,'Diego Mariano', '1990-01-01', 'Av. Ant. Carlos, 6673', 'BELO
```

```
HORIZONTE', 'MG', '01234567891'),
```

```
(DEFAULT,'Fliper Ama', '2017-01-01', 'Av. Ant. Carlos, 6600', 'BELO  
HORIZONTE', 'MG', '11111111111'),
```

```
(DEFAULT,'Pedro Martins', '1997-02-13', 'Av. Brasil, 1000', 'CABO FRIO', 'RJ',  
'22222222222'),
```

```
(DEFAULT,'REGINA CAZÉ', '1920-01-01', 'Rua do Mar', 'SALVADOR', 'BA',  
'33333333333');
```

### **# POVOANDO TABELAS CURSOS**

```
INSERT INTO cursos VALUES
```

```
(DEFAULT, "Codeigniter"),
```

```
(DEFAULT, "Python"),
```

```
(DEFAULT, "MySQL");
```

### **# POVOANDO TABELAS ALUNOS\_CURSOS**

```
INSERT INTO alunos_cursos VALUES
```

```
(DEFAULT, 1, 1), # Pedro (id_aluno = 1) está inscrito em Codeigniter (id_curso  
= 1)
```

```
(DEFAULT, 1, 2), # Pedro (id_aluno = 1) está inscrito em Python (id_curso = 2)
```

```
(DEFAULT, 2, 1), # Diego (id_aluno = 2) está inscrito em Codeigniter (id_curso  
= 1)
```

```
(DEFAULT, 2, 3), # Diego (id_aluno = 1) está inscrito em Mysql (id_curso = 3)
```

```
(DEFAULT, 3, 1), # Fliper (id_aluno = 3) está inscrito em Codeigniter (id_curso  
= 1)
```

```
(DEFAULT, 3, 2), # Fliper (id_aluno = 3) está inscrito em Python (id_curso = 2)
```



```
(DEFAULT, 4, 1), # Ricardo (id_aluno = 1) está inscrito em Codeigniter  
(id_curso = 1)
```

```
(DEFAULT, 5, 1); # Regina (id_aluno = 1) está inscrito em Codeigniter  
(id_curso = 1)
```

## # POVOANDO TABELAS NOTAS

```
INSERT INTO
```

```
notas VALUES
```

```
(DEFAULT, 1, 'Prova 1', 28.0), # Pedro fez a atividade Prova 1 no Codeigniter  
e tirou 28.0
```

```
(DEFAULT, 1, 'Prova 2', 25.0), # Pedro fez a atividade Prova 2 no Codeigniter  
e tirou 25.0
```

```
(DEFAULT, 2, 'Prova 2', 20.0), # Pedro fez a atividade Prova 2 no Python e  
tirou 20.0
```

```
(DEFAULT, 2, 'Prova 2', 20.0), # Pedro fez a atividade Exercício 2 no Python e  
tirou 10.0
```

```
(DEFAULT, 3, 'Prova 1', 25.0), # Diego fez a atividade Prova 1 no Codeigniter e  
tirou 25.0
```

```
(DEFAULT, 5, 'Prova 1', 28.0), # Fliper fez a atividade Prova 1 no Codeigniter e  
tirou 28.0
```

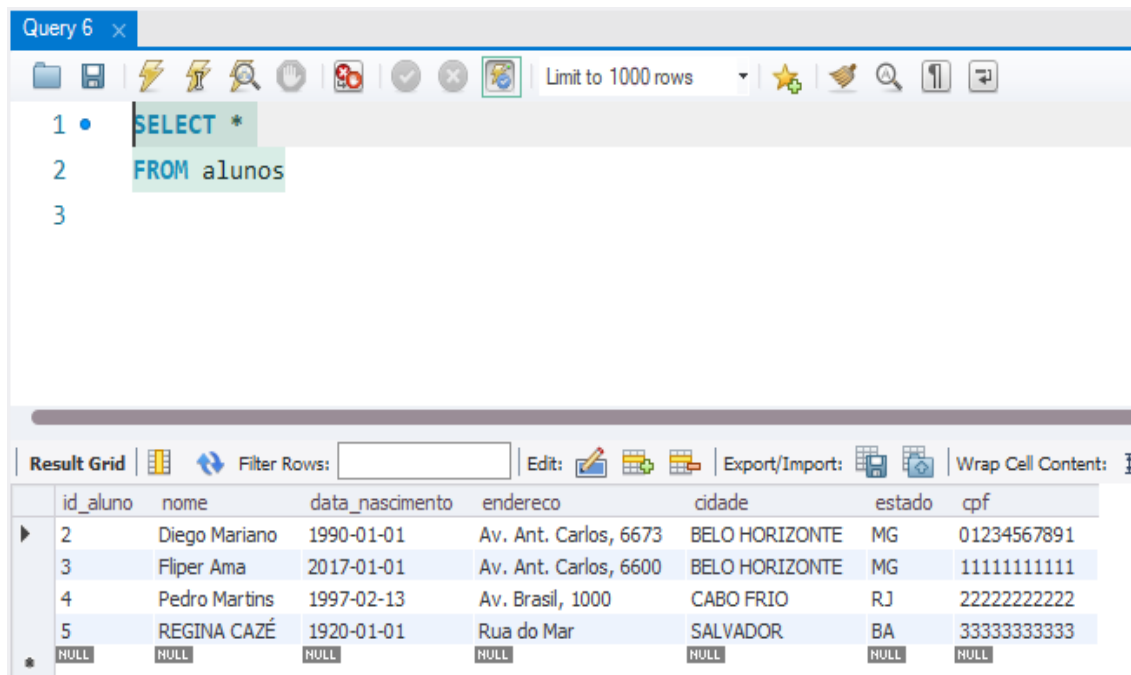
```
(DEFAULT, 6, 'Exercicio 2', 12.0); # Fliper fez a atividade Exercicio 2 no Python  
e tirou 12.0
```

## Realizando consultas nas Tabelas

1ª Consulta (Todos os alunos)

```
SELECT *
```

```
FROM alunos
```



Query 6

```
1 • SELECT *
2 FROM alunos
3
```

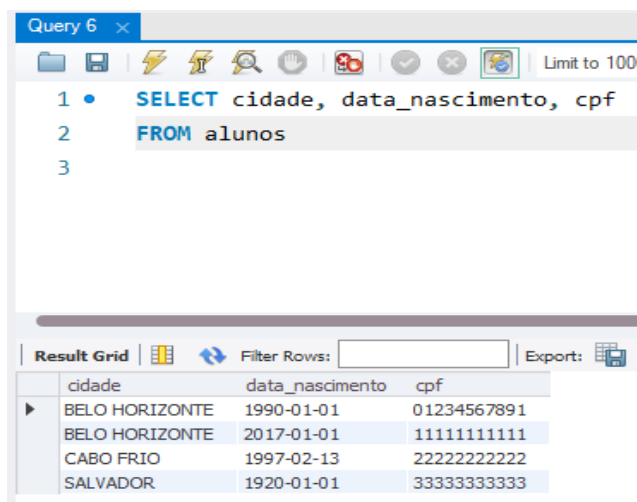
Result Grid

	id_aluno	nome	data_nascimento	endereco	cidade	estado	cpf
▶	2	Diego Mariano	1990-01-01	Av. Ant. Carlos, 6673	BELO HORIZONTE	MG	01234567891
	3	Filiper Ama	2017-01-01	Av. Ant. Carlos, 6600	BELO HORIZONTE	MG	11111111111
	4	Pedro Martins	1997-02-13	Av. Brasil, 1000	CABO FRIO	RJ	22222222222
	5	REGINA CAZÉ	1920-01-01	Rua do Mar	SALVADOR	BA	33333333333
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2ª Consulta (Cidade, Data de Nascimento e CPF dos Alunos)

```
SELECT cidade, data_nascimento, cpf
```

```
FROM alunos
```



Query 6

```
1 • SELECT cidade, data_nascimento, cpf
2 FROM alunos
3
```

Result Grid

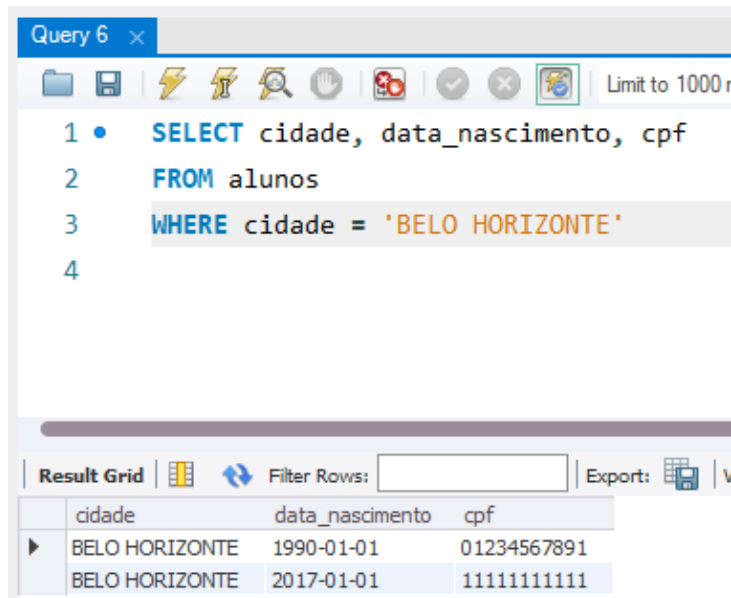
	cidade	data_nascimento	cpf
▶	BELO HORIZONTE	1990-01-01	01234567891
	BELO HORIZONTE	2017-01-01	11111111111
	CABO FRIO	1997-02-13	22222222222
	SALVADOR	1920-01-01	33333333333

### 3ª Consulta (Alunos da cidade de Belo Horizonte)

```
SELECT cidade, data_nascimento, cpf
```

```
FROM alunos
```

```
WHERE cidade = 'BELO HORIZONTE'
```



The screenshot shows a SQL query editor window titled "Query 6". The query is as follows:

```
1 • SELECT cidade, data_nascimento, cpf
2 FROM alunos
3 WHERE cidade = 'BELO HORIZONTE'
4
```

Below the query editor, the "Result Grid" is displayed, showing the results of the query. The grid has four columns: cidade, data\_nascimento, and cpf. There are two rows of data.

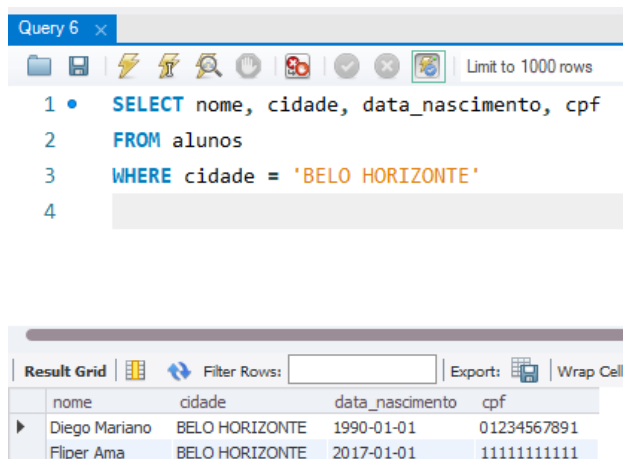
	cidade	data_nascimento	cpf
▶	BELO HORIZONTE	1990-01-01	01234567891
	BELO HORIZONTE	2017-01-01	11111111111

### 4ª Consulta (Nomes dos alunos da cidade de Belo Horizonte)

```
SELECT nome, cidade, data_nascimento, cpf
```

```
FROM alunos
```

```
WHERE cidade = 'BELO HORIZONTE'
```



The screenshot shows a SQL query editor window titled "Query 6". The query is as follows:

```
1 • SELECT nome, cidade, data_nascimento, cpf
2 FROM alunos
3 WHERE cidade = 'BELO HORIZONTE'
4
```

Below the query editor, the "Result Grid" is displayed, showing the results of the query. The grid has five columns: nome, cidade, data\_nascimento, and cpf. There are two rows of data.

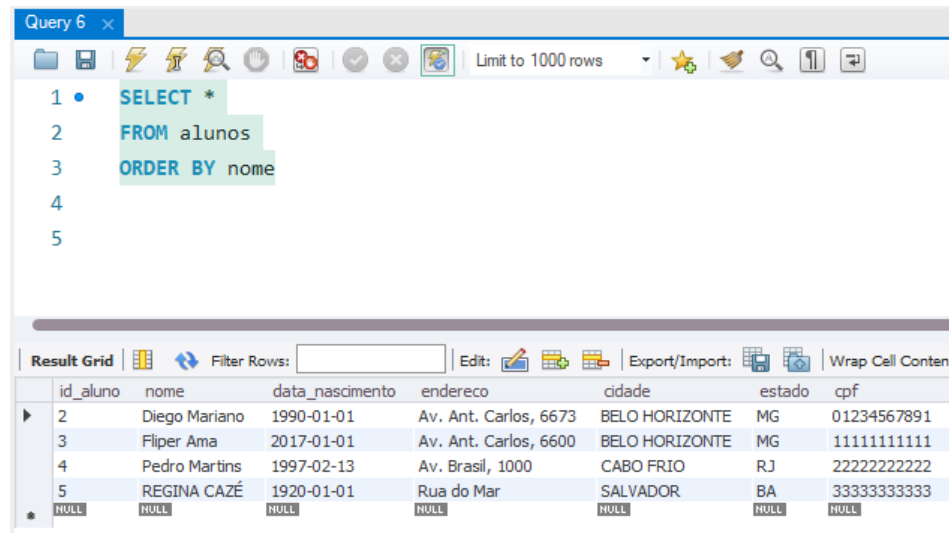
	nome	cidade	data_nascimento	cpf
▶	Diego Mariano	BELO HORIZONTE	1990-01-01	01234567891
	Filiper Ama	BELO HORIZONTE	2017-01-01	11111111111

### 5ª Consulta (Ordem alfabética do nome)

```
SELECT *
```

```
FROM alunos
```

```
ORDER BY nome
```



Query 6

```
1 • SELECT *
2 FROM alunos
3 ORDER BY nome
4
5
```

Result Grid

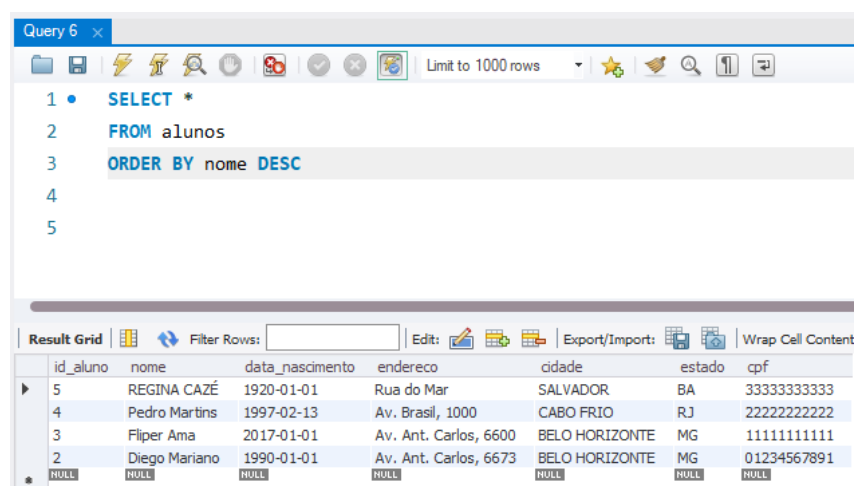
	id_aluno	nome	data_nascimento	endereco	cidade	estado	cpf
▶	2	Diego Mariano	1990-01-01	Av. Ant. Carlos, 6673	BELO HORIZONTE	MG	01234567891
	3	Fliper Ama	2017-01-01	Av. Ant. Carlos, 6600	BELO HORIZONTE	MG	11111111111
	4	Pedro Martins	1997-02-13	Av. Brasil, 1000	CABO FRIO	RJ	22222222222
	5	REGINA CAZÉ	1920-01-01	Rua do Mar	SALVADOR	BA	33333333333
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

### 6ª Consulta (Ordem alfabética descendente do nome)

```
SELECT *
```

```
FROM alunos
```

```
ORDER BY nome DESC
```



Query 6

```
1 • SELECT *
2 FROM alunos
3 ORDER BY nome DESC
4
5
```

Result Grid

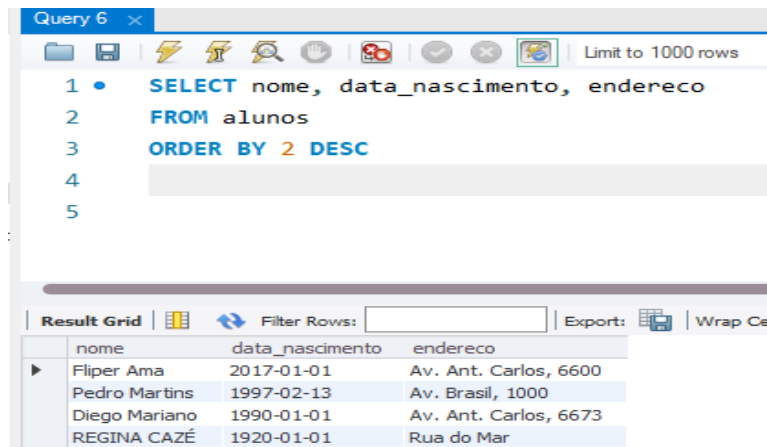
	id_aluno	nome	data_nascimento	endereco	cidade	estado	cpf
▶	5	REGINA CAZÉ	1920-01-01	Rua do Mar	SALVADOR	BA	33333333333
	4	Pedro Martins	1997-02-13	Av. Brasil, 1000	CABO FRIO	RJ	22222222222
	3	Fliper Ama	2017-01-01	Av. Ant. Carlos, 6600	BELO HORIZONTE	MG	11111111111
	2	Diego Mariano	1990-01-01	Av. Ant. Carlos, 6673	BELO HORIZONTE	MG	01234567891
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

7ª Consulta (Ordem descendente do segundo campo do Select)

```
SELECT nome, data_nascimento, endereco
```

```
FROM alunos
```

```
ORDER BY 2 DESC
```



Query 6

```
1 • SELECT nome, data_nascimento, endereco
2   FROM alunos
3   ORDER BY 2 DESC
4
5
```

Result Grid

	nome	data_nascimento	endereco
▶	Filiper Ama	2017-01-01	Av. Ant. Carlos, 6600
	Pedro Martins	1997-02-13	Av. Brasil, 1000
	Diego Mariano	1990-01-01	Av. Ant. Carlos, 6673
	REGINA CAZÉ	1920-01-01	Rua do Mar

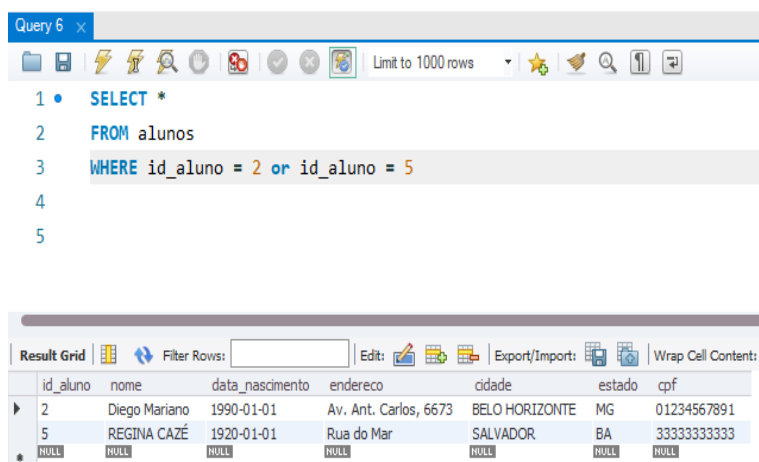
## Operadores Lógicos e Racionais

Uma ou outra condição tem que ser verdadeira para retornar o resultado do *Select*:

```
SELECT *
```

```
FROM alunos
```

```
WHERE id_aluno = 2 or id_aluno = 5
```



Query 6

```
1 • SELECT *
2   FROM alunos
3   WHERE id_aluno = 2 or id_aluno = 5
4
5
```

Result Grid

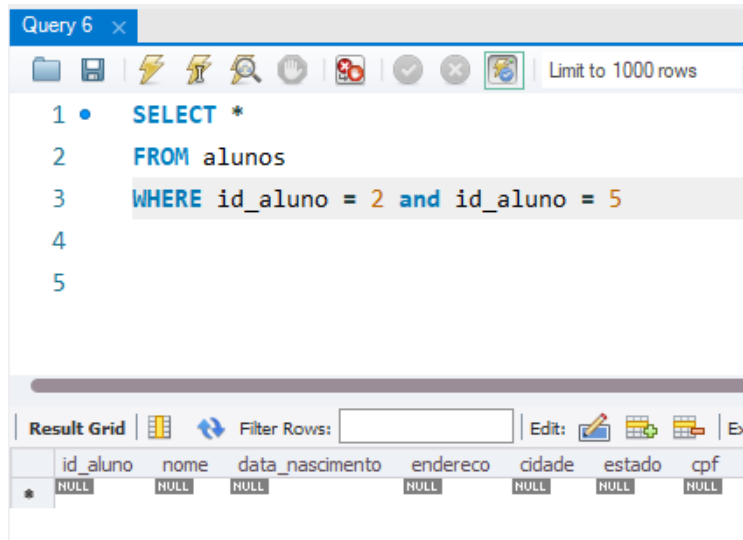
	id_aluno	nome	data_nascimento	endereco	cidade	estado	cpf
▶	2	Diego Mariano	1990-01-01	Av. Ant. Carlos, 6673	BELO HORIZONTE	MG	01234567891
	5	REGINA CAZÉ	1920-01-01	Rua do Mar	SALVADOR	BA	33333333333
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

As duas condições têm que ser verdadeiras para retornar o resultado do *Select*:

```
SELECT *
```

```
FROM alunos
```

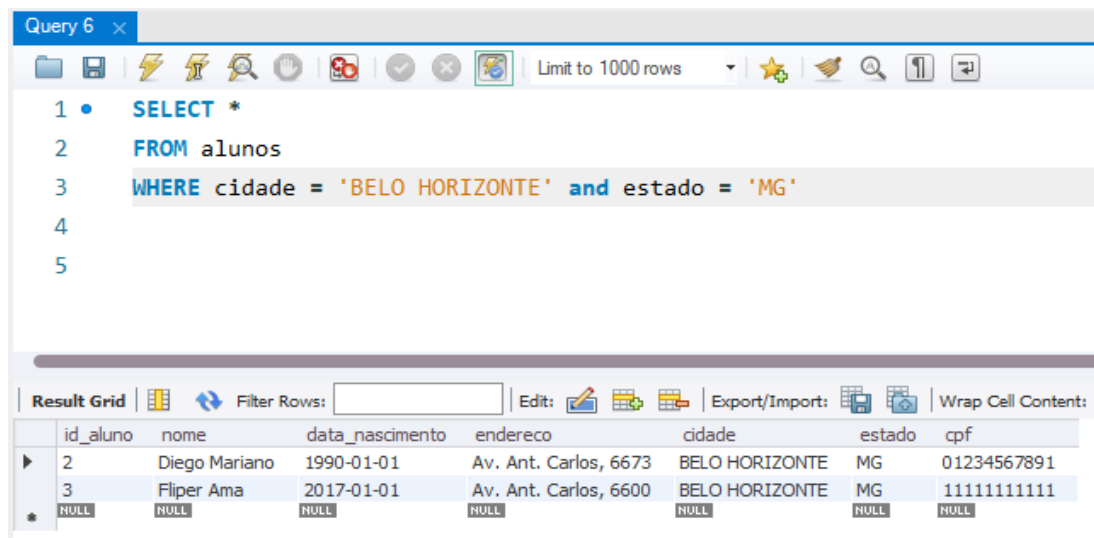
```
WHERE id_aluno = 2 and id_aluno = 5
```



```
SELECT *
```

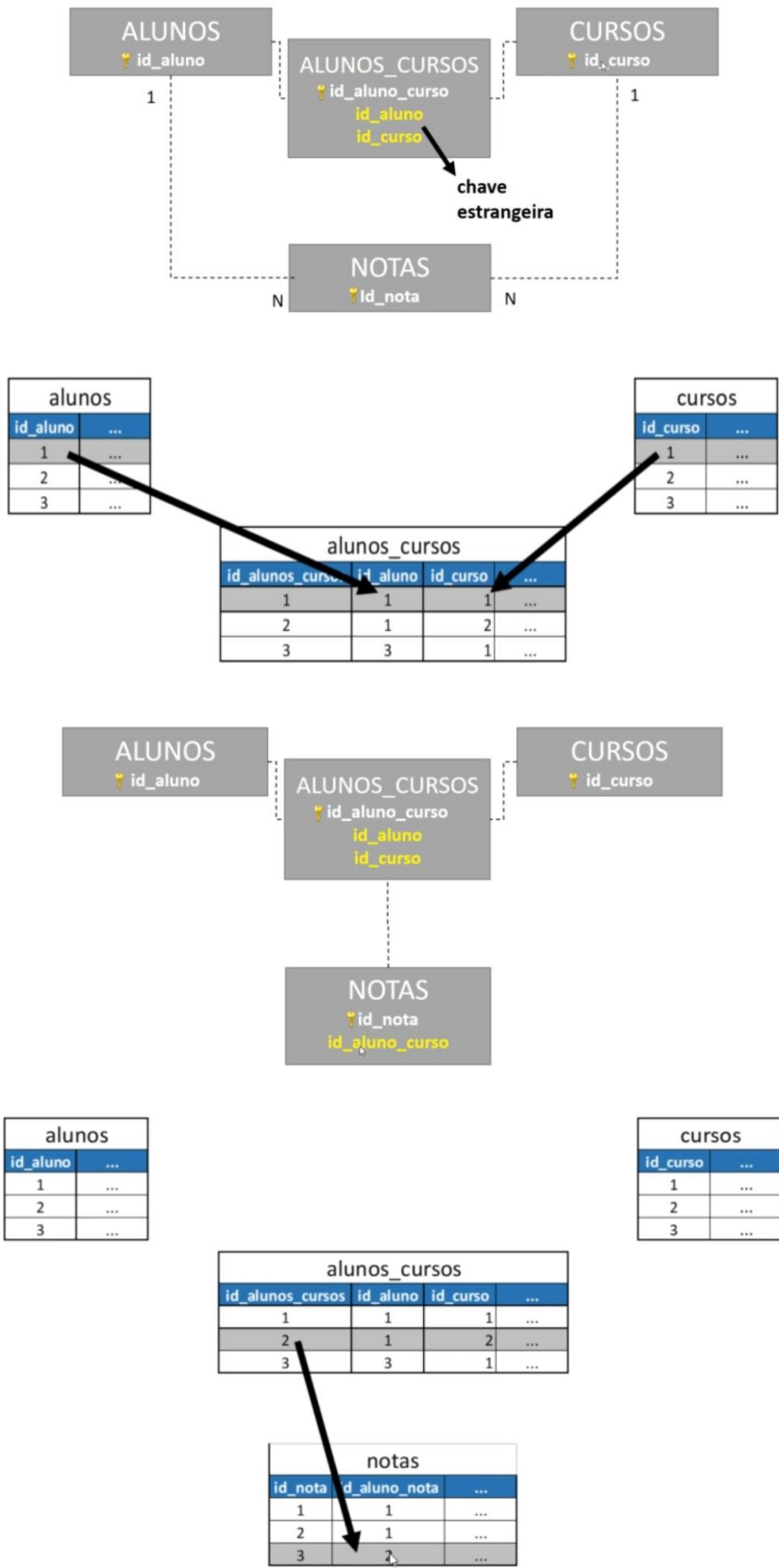
```
FROM alunos
```

```
WHERE cidade = 'BELO HORIZONTE' and estado = 'MG'
```





Assim, quando se tem uma cardinalidade n para n, como a cardinalidade de aluno para curso, precisa-se criar uma relação criando uma nova tabela, com seu próprio id e as chaves estrangeiras relacionando-se entre si.





# Criando Chaves Estrangeiras

## FK para ALUNOS\_CURSOS

Navigator

SCHEMAS

Filter objects

escola\_curso

Tables

alunos

cursos

notas

Views

Stored Procedures

Functions

sakila

sys

udemy

world

Administration

Schemas

Information

Schema: escola\_curso

alunos

alunos\_cursos - Table

Table Name: alunos\_cursos

Schema: escola\_curso

Charset/Collation: Default Charset

Default Collation

Engine: InnoDB

Comments:

Foreign Key Name	Referenced Table	Column	Referenced Column
fk_id_aluno	escola_curso, alunos	<input type="checkbox"/> id_aluno_curso	
fk_id_curso	escola_curso, cursos	<input checked="" type="checkbox"/> id_aluno	id_aluno
		<input type="checkbox"/> id_curso	

Columns

Indexes

Foreign Keys

Triggers

Partitioning

Options

Navigator

SCHEMAS

Filter objects

escola\_curso

Tables

alunos

cursos

notas

Views

Stored Procedures

Functions

sakila

sys

udemy

world

Administration

Schemas

Information

Schema: escola\_curso

alunos

alunos\_cursos - Table

Table Name: alunos\_cursos

Schema: escola\_curso

Charset/Collation: Default Charset

Default Collation

Engine: InnoDB

Comments:

Foreign Key Name	Referenced Table	Column	Referenced Column
fk_id_aluno	escola_curso, alunos	<input type="checkbox"/> id_aluno_curso	
fk_id_curso	escola_curso, cursos	<input type="checkbox"/> id_aluno	
		<input checked="" type="checkbox"/> id_curso	id_cursos

Columns

Indexes

Foreign Keys

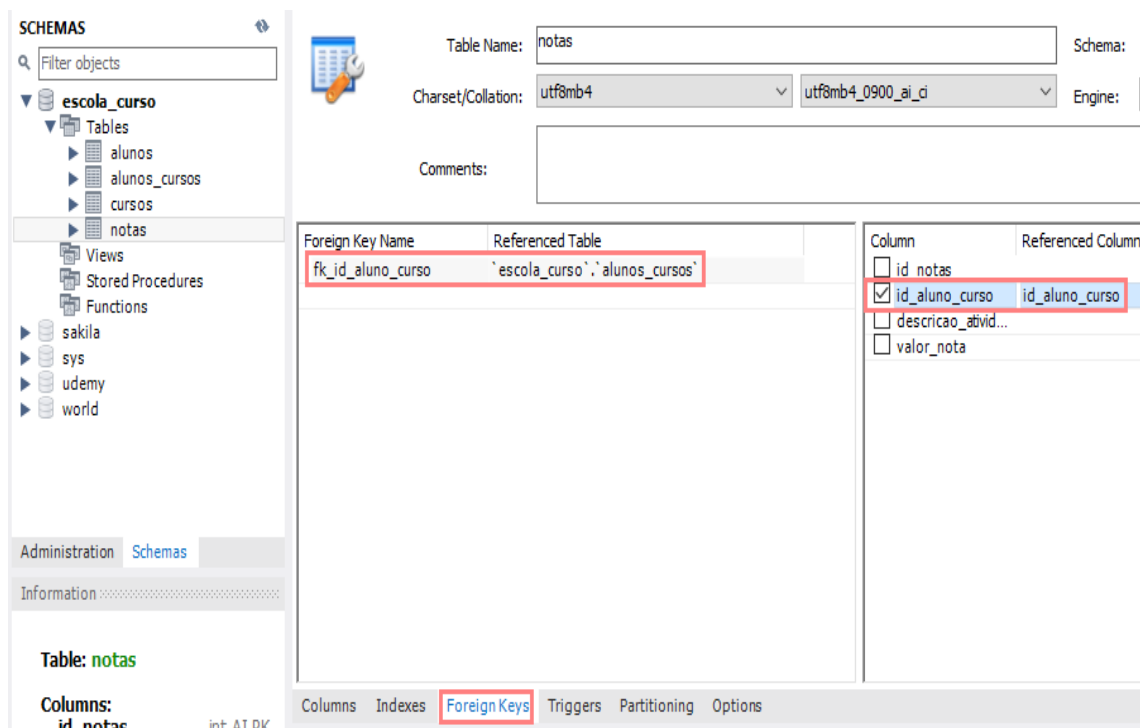
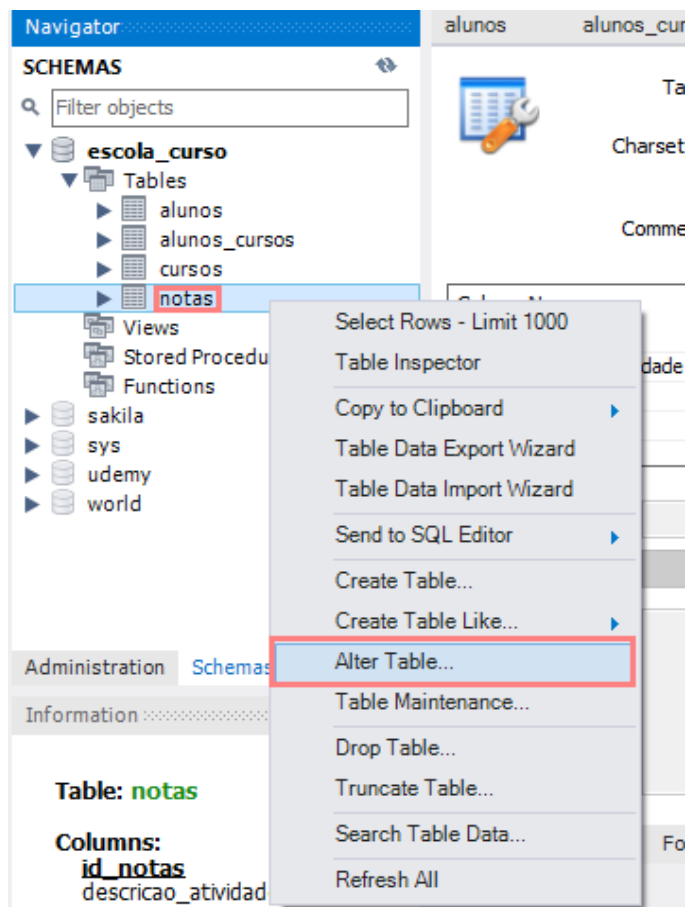
Triggers

Partitioning

Options

Sumário

## Foreign Key para notas



## Criando outras Tabelas com suas FK

### # CRIANDO TABELA ALUNOS

```
CREATE TABLE `alunos` (  
  
  `id_aluno` int NOT NULL AUTO_INCREMENT,  
  
  `nome` varchar(100) NOT NULL,  
  
  `data_nascimento` date NOT NULL,  
  
  `endereco` varchar(255) NOT NULL,  
  
  `cidade` varchar(100) NOT NULL,  
  
  `estado` varchar(2) NOT NULL,  
  
  `cpf` varchar(11) NOT NULL,  
  
  PRIMARY KEY (`id_aluno`)  
  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

### # CRIANDO TABELA CURSOS

```
CREATE TABLE `cursos` (  
  
  `id_curso` int NOT NULL AUTO_INCREMENT,  
  
  `nome` varchar(100) NOT NULL,  
  
  PRIMARY KEY (`id_curso`)  
  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

## # CRIANDO TABELA ALUNOS\_CURSOS COM FK

```
CREATE TABLE `alunos_cursos` (  
  
  `id_aluno_curso` int NOT NULL AUTO_INCREMENT,  
  
  `id_aluno` int NOT NULL,  
  
  `id_curso` int NOT NULL,  
  
  PRIMARY KEY (`id_aluno_curso`),  
  
  KEY `fk_alunos_cursos_1_idx` (`id_aluno`),  
  
  KEY `fk_alunos_cursos_2_idx` (`id_curso`),  
  
  CONSTRAINT `fk_alunos_cursos_1` FOREIGN KEY (`id_aluno`) REFERENCES `alunos` (`id_aluno`) ON DELETE NO ACTION ON UPDATE NO ACTION,  
  
  CONSTRAINT `fk_alunos_cursos_2` FOREIGN KEY (`id_curso`) REFERENCES `cursos` (`id_curso`) ON DELETE NO ACTION ON UPDATE NO ACTION  
  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

## # CRIANDO TABELA NOTAS COM FK

```
CREATE TABLE `notas` (  
  
  `id_nota` int NOT NULL AUTO_INCREMENT,  
  
  `id_aluno_curso` int NOT NULL,  
  
  `descricao_atividade` varchar(100) NOT NULL,
```

```
`vlr_nota` decimal(5,2) NOT NULL,
```

```
PRIMARY KEY (`id_nota`),
```

```
KEY `fk_notas_1_idx` (`id_aluno_curso`),
```

```
CONSTRAINT `fk_notas_1` FOREIGN KEY (`id_aluno_curso`) REFERENCES  
`alunos_cursos` (`id_aluno_curso`) ON DELETE NO ACTION ON UPDATE NO  
ACTION
```

```
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

## Joins

### # JOIN ENTRE ALUNOS E ALUNOS\_CURSOS

```
SELECT A.id_aluno, A.nome, C.id_curso, C.nome
```

```
FROM alunos A,
```

```
    cursos C,
```

```
    alunos_cursos AC
```

```
WHERE A.id_aluno = AC.id_aluno and C.id_curso = AC.id_curso
```

The screenshot shows a MySQL database interface. On the left, the 'SCHEMAS' panel displays a tree view of the database structure, including tables like 'alunos', 'alunos\_cursos', 'cursos', and 'notas'. The main query editor on the right contains the following SQL query:

```
1 SELECT A.id_aluno, A.nome, C.id_curso, C.nome  
2 FROM alunos A,  
3     cursos C,  
4     alunos_cursos AC  
5 WHERE A.id_aluno = AC.id_aluno and C.id_curso = AC.id_curso
```

Below the query editor, the 'Result Grid' displays the results of the query. The results are as follows:

	id_aluno	nome	id_curso	nome
1	Pedro Martins	1	Codeigniter	
2	Diego Mariano	1	Codeigniter	
3	Filipe Ama	1	Codeigniter	
4	Pedro Martins	1	Codeigniter	
5	REGINA CAZÉ	1	Codeigniter	
1	Pedro Martins	2	Python	
3	Filipe Ama	2	Python	
2	Diego Mariano	3	MySQL	

## # JOIN ADICIONANDO AS NOTAS

```
SELECT A.nome, C.nome, N.descricao_atividade, N.vlr_nota
```

```
FROM alunos A,
```

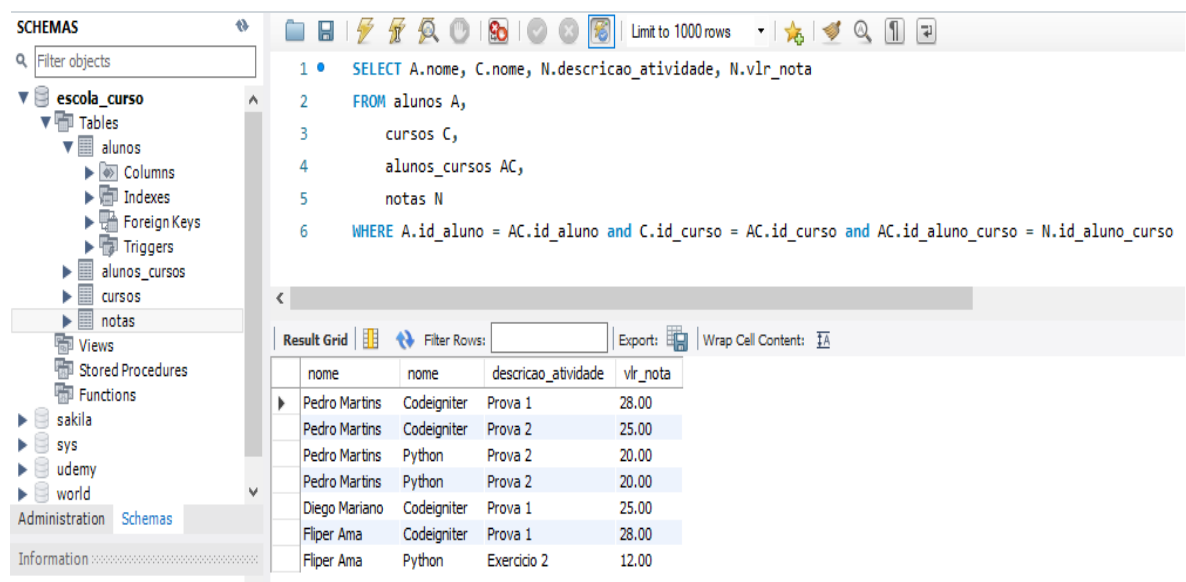
```
    cursos C,
```

```
    alunos_cursos AC,
```

```
    notas N
```

```
WHERE A.id_aluno = AC.id_aluno and C.id_curso = AC.id_curso and
```

```
AC.id_aluno_curso = N.id_aluno_curso
```



The screenshot shows a MySQL database interface. On the left, the 'SCHEMAS' panel displays a tree view of the database structure, including tables like 'alunos', 'cursos', 'alunos\_cursos', and 'notas'. The main area shows a SQL query being executed:

```
1 • SELECT A.nome, C.nome, N.descricao_atividade, N.vlr_nota
2 FROM alunos A,
3     cursos C,
4     alunos_cursos AC,
5     notas N
6 WHERE A.id_aluno = AC.id_aluno and C.id_curso = AC.id_curso and AC.id_aluno_curso = N.id_aluno_curso
```

Below the query, the 'Result Grid' displays the following data:

nome	nome	descricao_atividade	vlr_nota
Pedro Martins	Codeigniter	Prova 1	28.00
Pedro Martins	Codeigniter	Prova 2	25.00
Pedro Martins	Python	Prova 2	20.00
Pedro Martins	Python	Prova 2	20.00
Diego Mariano	Codeigniter	Prova 1	25.00
Filiper Ama	Codeigniter	Prova 1	28.00
Filiper Ama	Python	Exercicio 2	12.00

## Estudo de Caso: Base de Dados de Filmes

Qual o melhor filme produzido entre 2007 e 2011?

Neste estudo de caso, usou-se MySQL para manipular uma base de dados de filmes produzidos entre 2007 e 2011. Usando MySQL, pode-se definir quais os melhores filmes produzidos avaliados pela crítica e pelo público. Também conheceu-se quais os filmes com maior custo de produção e quais custaram pouco.

A base de dados contém notas avaliadas pelo [ROTTEN TOMATOES](#). O Rotten Tomatoes é considerado um dos sites de recomendação mais confiáveis do mundo para entretenimento de qualidade.

A base de dados utilizada foi coletada no Kaggle:

<https://www.kaggle.com/nagenderp/movie-ratings/download>

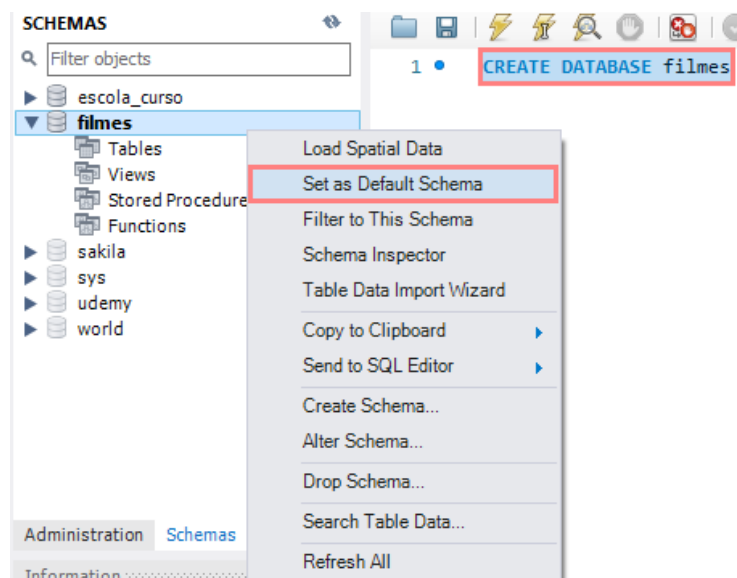
Perguntas para serem respondidas com o MySQL sobre a base de dados Filmes:

1. Quais são os 10 filmes mais apreciados pelo público?
2. Quais são os 10 filmes mais apreciados pela crítica especializada?
3. Quais são os 10 filmes mais odiados pelo público?
4. Quais são os 10 filmes mais odiados pela crítica especializada?
5. Qual filme com maior custo e qual filme com menor custo?
6. Qual a média da nota da crítica especializada?
7. Qual a média da nota do público?
8. Qual a média de custo de filmes?
9. Quantos filmes custaram mais do que o custo médio dos filmes da tabela?
10. Quais são os filmes com nota acima da média das notas dadas pela crítica especializada?
11. Quais são os filmes com nota acima da média das notas dadas pelo público? Quais os melhores?

12. Quais são os tipos de categoria (gêneros) existentes?
13. Quais são os gêneros com maior quantidade de filmes?
14. Qual gênero tem a mais alta média de custo?
15. Qual gênero tem a mais alta média de nota para o público?
16. Qual gênero tem a mais alta média de nota para a crítica especializada?
17. Quantos filmes foram produzidos por ano?
18. Qual ano foram produzidos mais filmes?
19. Qual gênero produziu mais filmes em um ano?
20. Qual o filme mais amado pela audiência e pelos especialistas ao mesmo tempo?

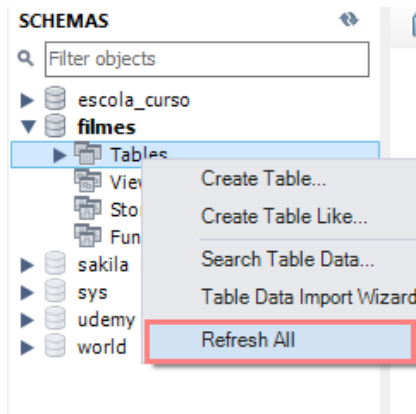
## Criando a Database Filmes

**CREATE DATABASE filmes**





Importante lembrar-se de executar um *Refresh All* sempre que sentir necessidade



## Criando a Tabela Filmes

```
CREATE TABLE filmes(  
    id_filme int auto_increment primary key,  
    filme varchar(255),  
    genero varchar(255),  
    nota_especialistas int,  
    nota_audiencia int,  
    custo int,  
    ano int  
)
```

# Importando o arquivo .csv Filmes

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows the 'filmes' table under the 'Tables' folder. The main pane displays the 'Result Grid' for the query 'SELECT \* FROM filmes'. The grid shows columns: id\_filme, filme, genero, nota\_especialistas, nota\_audiencia, custo, and ano. The first row is highlighted, showing values for the first six columns. A red box highlights the 'Import records from an external file' button in the 'Export/Import' menu.

id_filme	filme	genero	nota_especialistas	nota_audiencia	custo	ano
1	(500) Days of Summer	Comedy	87	81	8	2009
2	10,000 B.C.	Adventure	9	44	105	2008
3	12 Rounds	Action	30	52	20	2009
4	127 Hours	Adventure	93	84	18	2010
5	17 Again	Comedy	55	70	20	2009
6	2012	Action	30	63	200	2009

The screenshot shows the 'Table Data Import' dialog box, 'Select Destination' tab. The 'Use existing table' option is selected, and the destination table is 'filmes.filmes'. The 'Create new table' option is also visible, with 'filmes' as the table name. The 'Truncate table before import' checkbox is unchecked.

Select destination table and additional options.

☒ Use existing table: filmes.filmes

☐ Create new table: filmes, filmes

☐ Truncate table before import

The screenshot shows the 'Table Data Import' dialog box, 'Configure Import Settings' tab. The 'Detected file format' is 'csv'. The 'Encoding' is set to 'latin1 (iso8859-1)'. A red box highlights the 'Encoding' dropdown menu.

Detected file format: csv

Encoding: latin1 (iso8859-1)

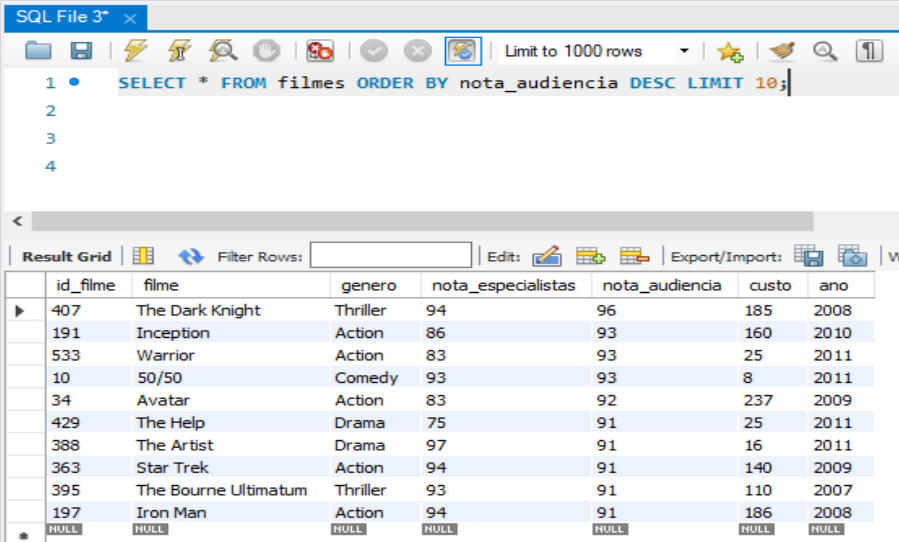
The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows the 'filmes' table under the 'Tables' folder. The main pane displays the 'Result Grid' for the query 'SELECT \* FROM filmes'. The grid shows columns: id\_filme, filme, genero, nota\_especialistas, nota\_audiencia, custo, and ano. The first row is highlighted, showing values for the first six columns. A red box highlights the 'Import records from an external file' button in the 'Export/Import' menu.

id_filme	filme	genero	nota_especialistas	nota_audiencia	custo	ano
1	(500) Days of Summer	Comedy	87	81	8	2009
2	10,000 B.C.	Adventure	9	44	105	2008
3	12 Rounds	Action	30	52	20	2009
4	127 Hours	Adventure	93	84	18	2010
5	17 Again	Comedy	55	70	20	2009
6	2012	Action	30	63	200	2009

# Análise da Base de Dados Filmes

1. Quais são os 10 filmes mais apreciados pelo público?

```
SELECT * FROM filmes ORDER BY nota_audiencia DESC LIMIT 10;
```



SQL File 3\* x

Limit to 1000 rows

1 • SELECT \* FROM filmes ORDER BY nota\_audiencia DESC LIMIT 10;

2

3

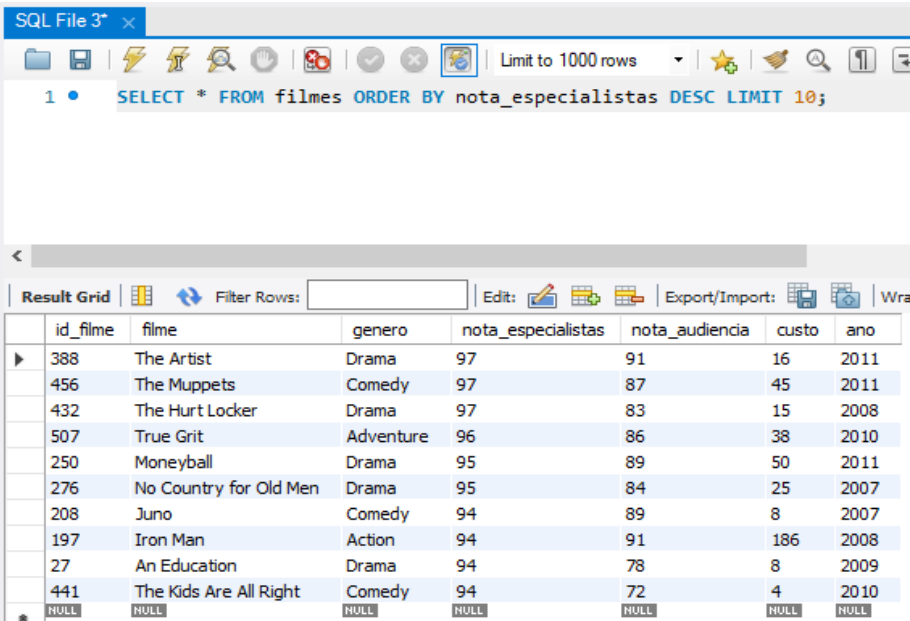
4

Result Grid

	id_filme	filme	genero	nota_especialistas	nota_audiencia	custo	ano
▶	407	The Dark Knight	Thriller	94	96	185	2008
	191	Inception	Action	86	93	160	2010
	533	Warrior	Action	83	93	25	2011
	10	50/50	Comedy	93	93	8	2011
	34	Avatar	Action	83	92	237	2009
	429	The Help	Drama	75	91	25	2011
	388	The Artist	Drama	97	91	16	2011
	363	Star Trek	Action	94	91	140	2009
	395	The Bourne Ultimatum	Thriller	93	91	110	2007
	197	Iron Man	Action	94	91	186	2008
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. Quais são os 10 filmes mais apreciados pela crítica especializada?

```
SELECT * FROM filmes ORDER BY nota_especialistas DESC LIMIT 10;
```



SQL File 3\* x

Limit to 1000 rows

1 • SELECT \* FROM filmes ORDER BY nota\_especialistas DESC LIMIT 10;

Result Grid

	id_filme	filme	genero	nota_especialistas	nota_audiencia	custo	ano
▶	388	The Artist	Drama	97	91	16	2011
	456	The Muppets	Comedy	97	87	45	2011
	432	The Hurt Locker	Drama	97	83	15	2008
	507	True Grit	Adventure	96	86	38	2010
	250	Moneyball	Drama	95	89	50	2011
	276	No Country for Old Men	Drama	95	84	25	2007
	208	Juno	Comedy	94	89	8	2007
	197	Iron Man	Action	94	91	186	2008
	27	An Education	Drama	94	78	8	2009
	441	The Kids Are All Right	Comedy	94	72	4	2010
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

3. Quais são os 10 filmes mais odiados pelo público?

```
SELECT * FROM filmes ORDER BY nota_audiencia ASC LIMIT 10;
```

SQL File 3\*

Limit to 1000 rows

```
1 SELECT * FROM filmes ORDER BY nota_audiencia ASC LIMIT 10;
```

Result Grid

	id_filme	filme	genero	nota_especialistas	nota_audiencia	custo	ano
▶	294	Paranormal Activity 2	Horror	0	0	3	2010
	355	Something Borrowed	Romance	0	0	35	2011
	352	Skyline	Action	16	19	10	2010
	368	Stone	Drama	49	20	22	2010
	427	The Haunting of Molly Hartley	Horror	3	22	5	2008
	504	Trespass	Thriller	11	24	35	2011
	205	Jonah Hex	Action	13	24	47	2010
	344	Shark Night 3D	Horror	16	25	25	2011
	214	Kit Kittredge: An American Girl	Drama	78	26	10	2008
	396	The Box	Drama	46	27	30	2009
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

4. Quais são os 10 filmes mais odiados pela crítica especializada?

**SELECT \* FROM filmes ORDER BY nota\_especialistas ASC LIMIT 10;**

SQL File 3\*

Limit to 1000 rows

```
1 SELECT * FROM filmes ORDER BY nota_especialistas ASC LIMIT 10;
```

Result Grid

	id_filme	filme	genero	nota_especialistas	nota_audiencia	custo	ano
▶	294	Paranormal Activity 2	Horror	0	0	3	2010
	355	Something Borrowed	Romance	0	0	35	2011
	286	One Missed Call	Horror	0	36	20	2008
	81	Daddy Day Camp	Comedy	1	63	6	2007
	243	Meet the Spartans	Comedy	2	31	30	2008
	115	Epic Movie	Comedy	2	38	20	2007
	94	Disaster Movie	Comedy	2	28	20	2008
	151	Good Luck Chuck	Comedy	3	61	25	2007
	427	The Haunting of Molly Hartley	Horror	3	22	5	2008
	201	Jack and Jill	Comedy	4	59	79	2011
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

5. Qual filme com maior custo e qual filme com menor custo?

**Maior Custo**

**# Opção 1**

**SELECT \* FROM filmes ORDER BY custo DESC LIMIT 1;**

**# Opção 2**

**SELECT \* FROM filmes WHERE custo = (SELECT max(custo) FROM filmes);**

SQL File 3\*

Limit to 1000 rows

```

1 # Opção 1
2 • SELECT * FROM filmes ORDER BY custo DESC LIMIT 1;
3 # Opção 2
4 • SELECT * FROM filmes WHERE custo = (SELECT max(custo) FROM filmes);

```

Result Grid

	id_filme	filme	genero	nota_especialistas	nota_audiencia	custo	ano
▶	305	Pirates of the Caribbean: At World's End	Action	45	74	300	2007
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

## Menor Custo

### # Opção 1

SELECT \* FROM filmes ORDER BY custo ASC LIMIT 1;

### # Opção 2

SELECT \* FROM filmes WHERE custo = (SELECT min(custo) FROM filmes)

LIMIT 1;

SQL File 3\*

Limit to 1000 rows

```

1 # Opção 1
2 • SELECT * FROM filmes ORDER BY custo ASC LIMIT 1;
3 # Opção 2
4 • SELECT * FROM filmes WHERE custo = (SELECT min(custo) FROM filmes) LIMIT 1;

```

Result Grid

	id_filme	filme	genero	nota_especialistas	nota_audiencia	custo	ano
▶	155	Greenberg	Comedy	75	40	0	2010
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

6. Qual a média da nota da crítica especializada?

SELECT AVG(nota\_especialistas) FROM filmes

SQL File 3\*

Limit to 1000 row

```

1 • SELECT AVG(nota_especialistas) FROM filmes
2 |

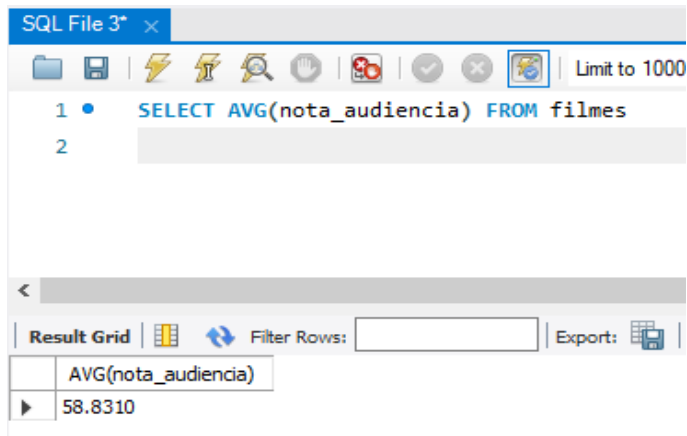
```

Result Grid

	AVG(nota_especialistas)
▶	47.4039

7. Qual a média da nota do público?

```
SELECT AVG(nota_audiencia) FROM filmes
```



The screenshot shows a SQL editor window titled "SQL File 3\*" with a toolbar and a "Limit to 1000" dropdown. The query editor contains the following SQL statement:

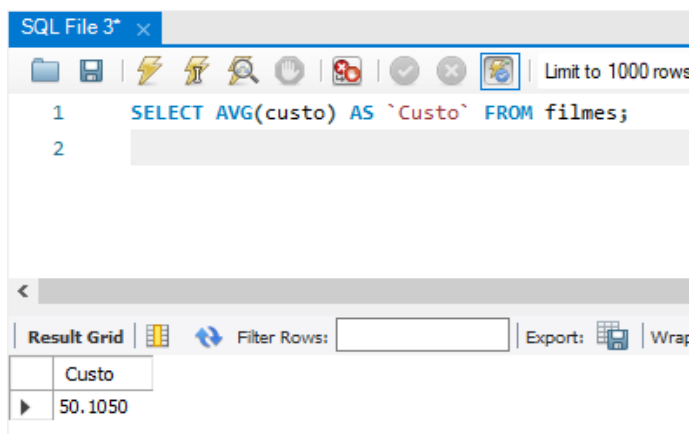
```
1 • SELECT AVG(nota_audiencia) FROM filmes
2
```

Below the editor is a "Result Grid" tab. The grid shows the following data:

AVG(nota_audiencia)
58.8310

8. Qual a média de custo de filmes?

```
SELECT AVG(custo) AS `Custo` FROM filmes;
```



The screenshot shows a SQL editor window titled "SQL File 3\*" with a toolbar and a "Limit to 1000 rows" dropdown. The query editor contains the following SQL statement:

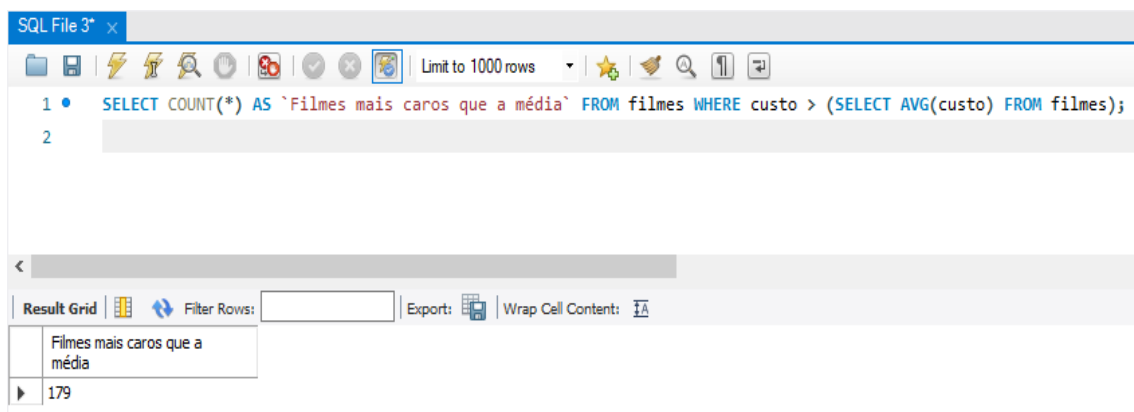
```
1 SELECT AVG(custo) AS `Custo` FROM filmes;
2
```

Below the editor is a "Result Grid" tab. The grid shows the following data:

Custo
50.1050

9. Quantos filmes custaram mais do que o custo médio dos filmes da tabela?

```
SELECT COUNT(*) AS `Filmes mais caros que a média` FROM filmes WHERE
custo > (SELECT AVG(custo) FROM filmes);
```



The screenshot shows a SQL editor window titled "SQL File 3\*" with a toolbar and a "Limit to 1000 rows" dropdown. The query editor contains the following SQL statement:

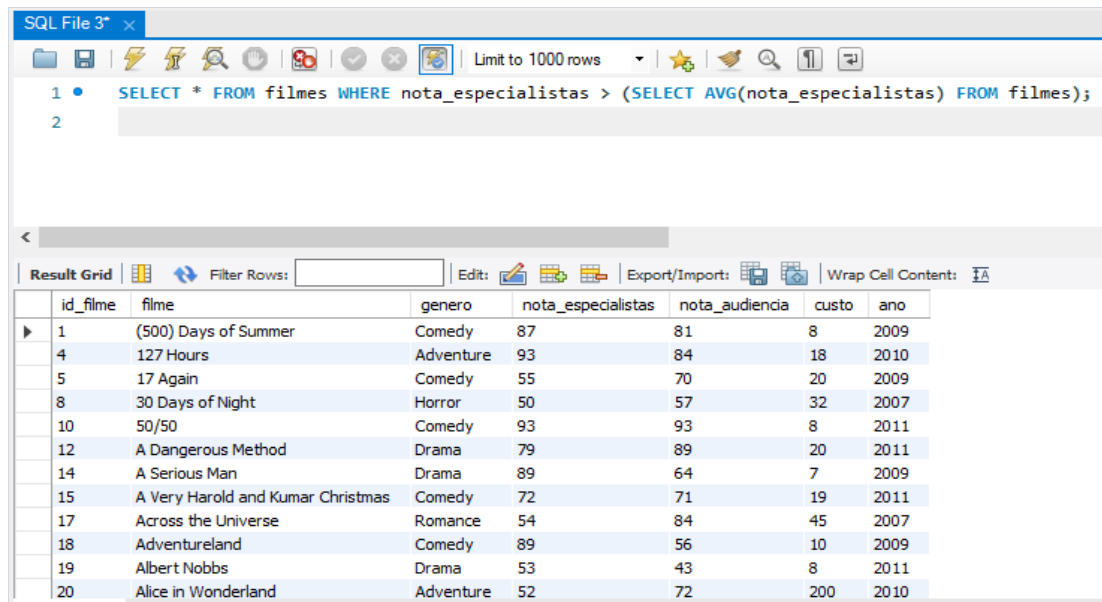
```
1 • SELECT COUNT(*) AS `Filmes mais caros que a média` FROM filmes WHERE custo > (SELECT AVG(custo) FROM filmes);
2
```

Below the editor is a "Result Grid" tab. The grid shows the following data:

Filmes mais caros que a média
179

10. Quais são os filmes com nota acima da média das notas dadas pela crítica especializada?

```
SELECT * FROM filmes WHERE nota_especialistas > (SELECT  
AVG(nota_especialistas) FROM filmes);
```



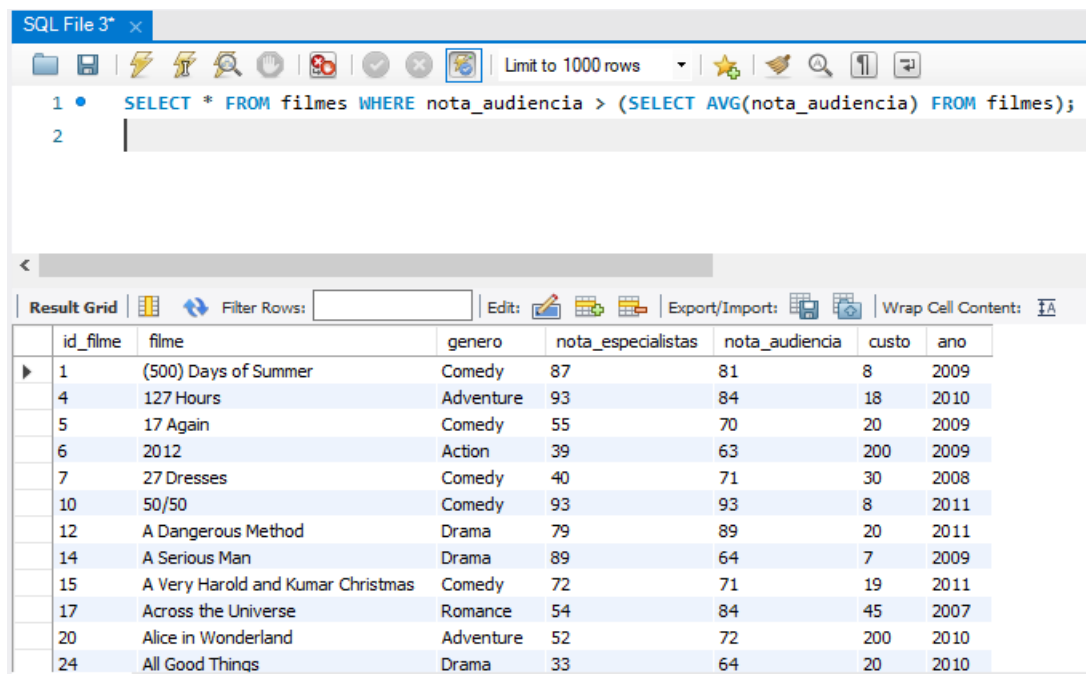
The screenshot shows a SQL query in a file named 'SQL File 3\*'. The query is: `SELECT * FROM filmes WHERE nota_especialistas > (SELECT AVG(nota_especialistas) FROM filmes);`. The result grid displays the following data:

	id_filme	filme	genero	nota_especialistas	nota_audiencia	custo	ano
▶	1	(500) Days of Summer	Comedy	87	81	8	2009
	4	127 Hours	Adventure	93	84	18	2010
	5	17 Again	Comedy	55	70	20	2009
	8	30 Days of Night	Horror	50	57	32	2007
	10	50/50	Comedy	93	93	8	2011
	12	A Dangerous Method	Drama	79	89	20	2011
	14	A Serious Man	Drama	89	64	7	2009
	15	A Very Harold and Kumar Christmas	Comedy	72	71	19	2011
	17	Across the Universe	Romance	54	84	45	2007
	18	Adventureland	Comedy	89	56	10	2009
	19	Albert Nobbs	Drama	53	43	8	2011
	20	Alice in Wonderland	Adventure	52	72	200	2010

11. Quais são os filmes com nota acima da média das notas dadas pelo público? Quais os melhores?

**Filmes com Nota Acima da Média**

```
SELECT * FROM filmes WHERE nota_audiencia > (SELECT  
AVG(nota_audiencia) FROM filmes);
```

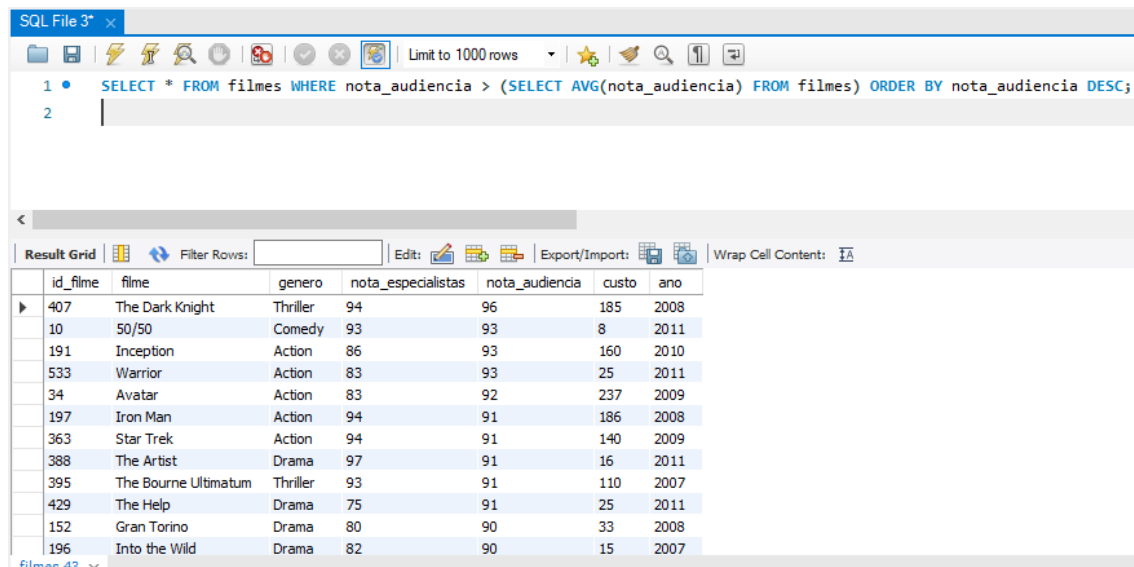


The screenshot shows a SQL query in a file named 'SQL File 3\*'. The query is: `SELECT * FROM filmes WHERE nota_audiencia > (SELECT AVG(nota_audiencia) FROM filmes);`. The result grid displays the following data:

	id_filme	filme	genero	nota_especialistas	nota_audiencia	custo	ano
▶	1	(500) Days of Summer	Comedy	87	81	8	2009
	4	127 Hours	Adventure	93	84	18	2010
	5	17 Again	Comedy	55	70	20	2009
	6	2012	Action	39	63	200	2009
	7	27 Dresses	Comedy	40	71	30	2008
	10	50/50	Comedy	93	93	8	2011
	12	A Dangerous Method	Drama	79	89	20	2011
	14	A Serious Man	Drama	89	64	7	2009
	15	A Very Harold and Kumar Christmas	Comedy	72	71	19	2011
	17	Across the Universe	Romance	54	84	45	2007
	20	Alice in Wonderland	Adventure	52	72	200	2010
	24	All Good Things	Drama	33	64	20	2010

## Melhores Filmes

```
SELECT * FROM filmes WHERE nota_audiencia > (SELECT  
AVG(nota_audiencia) FROM filmes) ORDER BY nota_audiencia DESC;
```



SQL File 3\* x

Limit to 1000 rows

```
1 • SELECT * FROM filmes WHERE nota_audiencia > (SELECT AVG(nota_audiencia) FROM filmes) ORDER BY nota_audiencia DESC;  
2 |
```

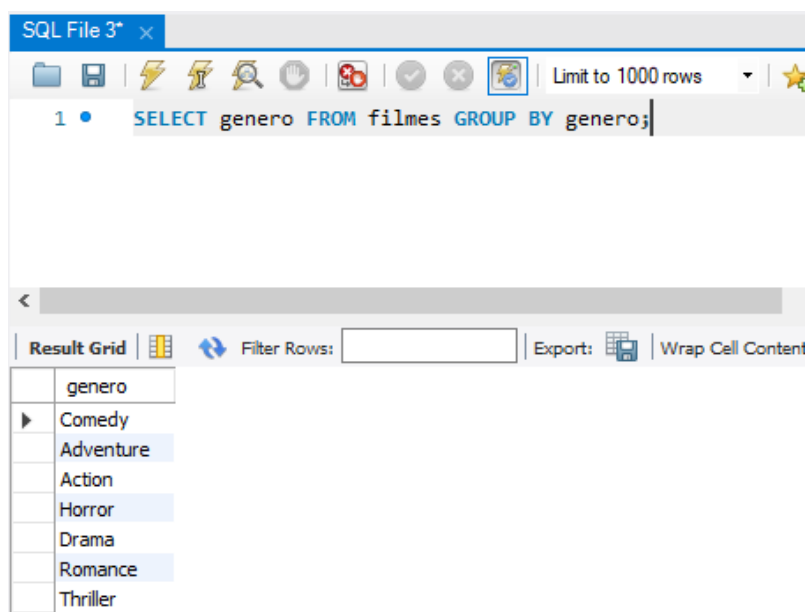
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	id_filme	filme	genero	nota_especialistas	nota_audiencia	custo	ano
▶	407	The Dark Knight	Thriller	94	96	185	2008
	10	50/50	Comedy	93	93	8	2011
	191	Inception	Action	86	93	160	2010
	533	Warrior	Action	83	93	25	2011
	34	Avatar	Action	83	92	237	2009
	197	Iron Man	Action	94	91	186	2008
	363	Star Trek	Action	94	91	140	2009
	388	The Artist	Drama	97	91	16	2011
	395	The Bourne Ultimatum	Thriller	93	91	110	2007
	429	The Help	Drama	75	91	25	2011
	152	Gran Torino	Drama	80	90	33	2008
	196	Into the Wild	Drama	82	90	15	2007

filmes 43 ▼

12. Quais são os tipos de categoria (gêneros) existentes?

```
SELECT genero FROM filmes GROUP BY genero;
```



SQL File 3\* x

Limit to 1000 rows

```
1 • SELECT genero FROM filmes GROUP BY genero;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

genero
Comedy
Adventure
Action
Horror
Drama
Romance
Thriller

13. Quais são os gêneros com maior quantidade de filmes?

```
SELECT genero, COUNT(genero) AS `Total` FROM filmes GROUP BY genero  
ORDER BY total DESC;
```



SQL File 3\* x

Limit to 1000 rows

```
1 • SELECT genero, COUNT(genero) AS `Total` FROM filmes GROUP BY genero ORDER BY total DESC;
```

Result Grid

genero	Total
Comedy	172
Action	154
Drama	101
Horror	49
Thriller	36
Adventure	29
Romance	21

14. Qual gênero tem a mais alta média de custo?

SELECT AVG(custo) as 'Comedy' from filmes WHERE genero = 'Comedy';

SELECT AVG(custo) as 'Action' from filmes WHERE genero = 'Action';

SELECT AVG(custo) as 'Drama' from filmes WHERE genero = 'Drama';

SELECT AVG(custo) as 'Horror' from filmes WHERE genero = 'Horror';

SELECT AVG(custo) as 'Thriller' from filmes WHERE genero = 'Thriller';

SELECT AVG(custo) as 'Adventure' from filmes WHERE genero = 'Adventure';

SELECT AVG(custo) as 'Romance' from filmes WHERE genero = 'Romance';

SQL File 3\* x

Limit to 1000 rows

```
1 • SELECT AVG(custo) as 'Comedy' from filmes WHERE genero = 'Comedy';
2 • SELECT AVG(custo) as 'Action' from filmes WHERE genero = 'Action';
3 • SELECT AVG(custo) as 'Drama' from filmes WHERE genero = 'Drama';
4 • SELECT AVG(custo) as 'Horror' from filmes WHERE genero = 'Horror';
5 • SELECT AVG(custo) as 'Thriller' from filmes WHERE genero = 'Thriller';
6 • SELECT AVG(custo) as 'Adventure' from filmes WHERE genero = 'Adventure';
7 • SELECT AVG(custo) as 'Romance' from filmes WHERE genero = 'Romance';
```

Result Grid

Action
84.6299

15. Qual gênero tem a mais alta média de nota para o público?

SELECT AVG(nota\_audiencia) as 'Comedy' from filmes WHERE genero = 'Comedy';

```
SELECT AVG(nota_audiencia) as 'Action' from filmes WHERE genero =  
'Action';
```

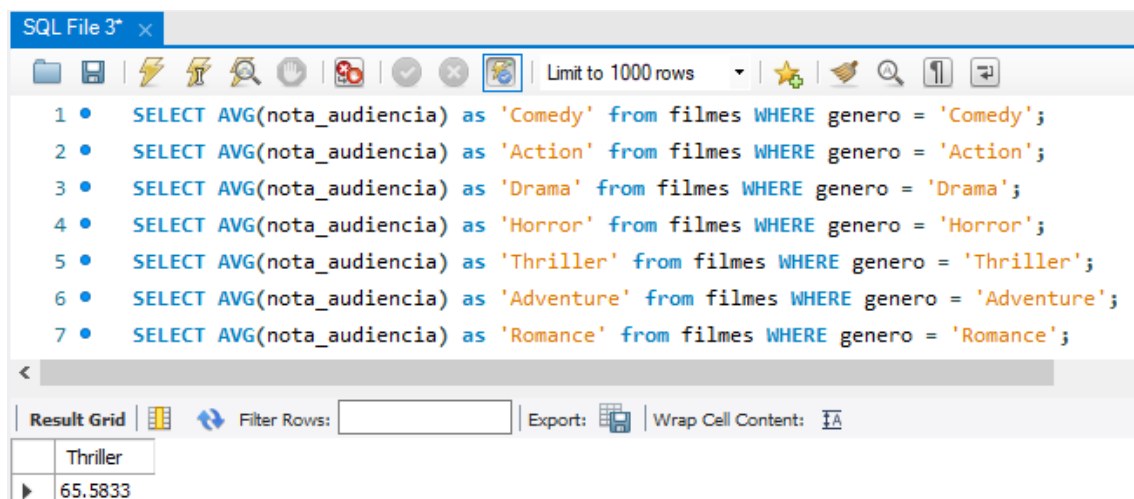
```
SELECT AVG(nota_audiencia) as 'Drama' from filmes WHERE genero =  
'Drama';
```

```
SELECT AVG(nota_audiencia) as 'Horror' from filmes WHERE genero =  
'Horror';
```

```
SELECT AVG(nota_audiencia) as 'Thriller' from filmes WHERE genero =  
'Thriller';
```

```
SELECT AVG(nota_audiencia) as 'Adventure' from filmes WHERE genero =  
'Adventure';
```

```
SELECT AVG(nota_audiencia) as 'Romance' from filmes WHERE genero =  
'Romance';
```



16. Qual gênero tem a mais alta média de nota para a crítica especializada?

```
SELECT AVG(nota_especialistas) as 'Comedy' from filmes WHERE genero =  
'Comedy';
```

```
SELECT AVG(nota_especialistas) as 'Action' from filmes WHERE genero =  
'Action';
```

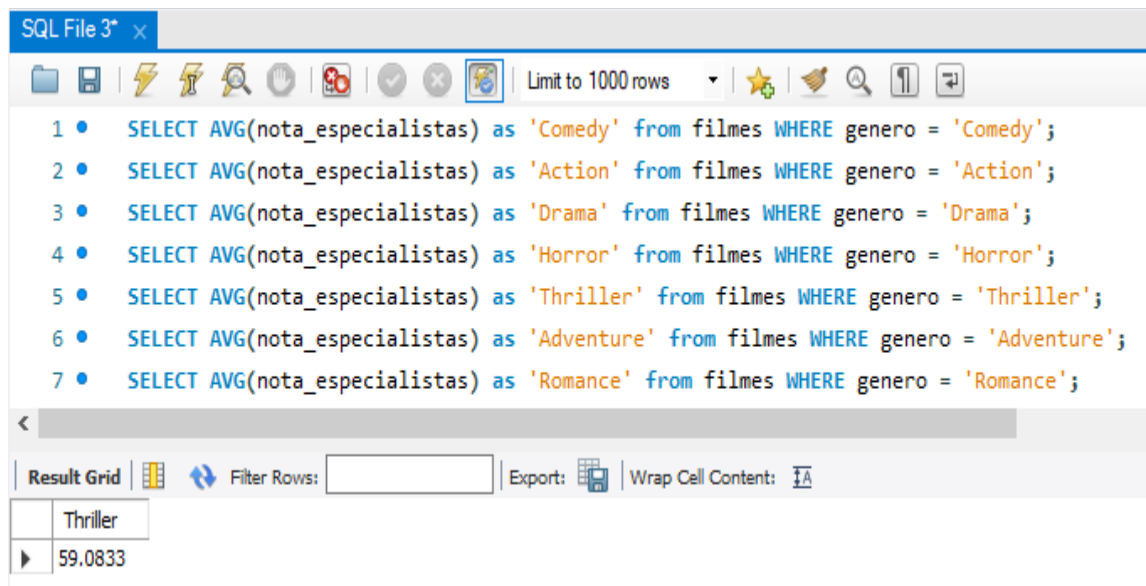
```
SELECT AVG(nota_especialistas) as 'Drama' from filmes WHERE genero =  
'Drama';
```

```
SELECT AVG(nota_especialistas) as 'Horror' from filmes WHERE genero =  
'Horror';
```

```
SELECT AVG(nota_especialistas) as 'Thriller' from filmes WHERE genero = 'Thriller';
```

```
SELECT AVG(nota_especialistas) as 'Adventure' from filmes WHERE genero = 'Adventure';
```

```
SELECT AVG(nota_especialistas) as 'Romance' from filmes WHERE genero = 'Romance';
```



The screenshot shows a SQL editor window titled "SQL File 3\*" with a toolbar and a list of 7 queries. The queries are:

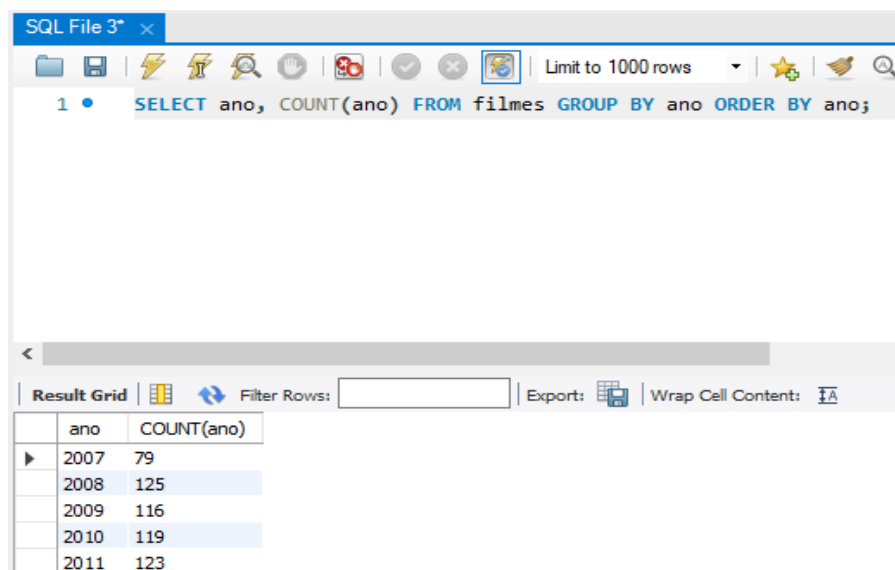
- 1 • SELECT AVG(nota\_especialistas) as 'Comedy' from filmes WHERE genero = 'Comedy';
- 2 • SELECT AVG(nota\_especialistas) as 'Action' from filmes WHERE genero = 'Action';
- 3 • SELECT AVG(nota\_especialistas) as 'Drama' from filmes WHERE genero = 'Drama';
- 4 • SELECT AVG(nota\_especialistas) as 'Horror' from filmes WHERE genero = 'Horror';
- 5 • SELECT AVG(nota\_especialistas) as 'Thriller' from filmes WHERE genero = 'Thriller';
- 6 • SELECT AVG(nota\_especialistas) as 'Adventure' from filmes WHERE genero = 'Adventure';
- 7 • SELECT AVG(nota\_especialistas) as 'Romance' from filmes WHERE genero = 'Romance';

Below the queries, the "Result Grid" is displayed for the 'Thriller' query, showing a single row with the value 59.0833.

Thriller
59.0833

17. Quantos filmes foram produzidos por ano?

```
SELECT ano, COUNT(ano) FROM filmes GROUP BY ano ORDER BY ano;
```



The screenshot shows a SQL editor window titled "SQL File 3\*" with a toolbar and a single query:

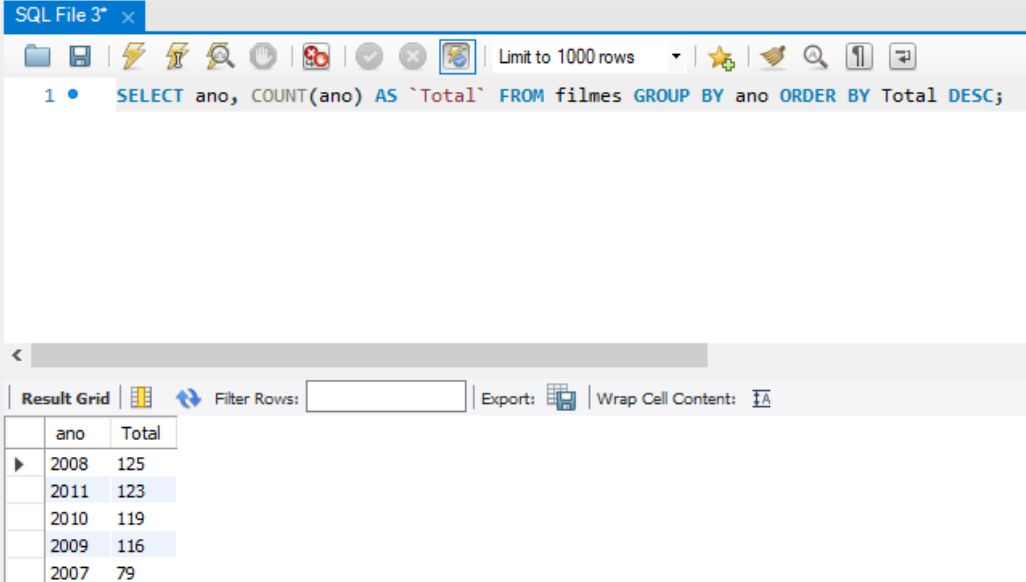
- 1 • SELECT ano, COUNT(ano) FROM filmes GROUP BY ano ORDER BY ano;

Below the query, the "Result Grid" is displayed, showing the results of the query:

ano	COUNT(ano)
2007	79
2008	125
2009	116
2010	119
2011	123

18. Qual ano foram produzidos mais filmes?

```
SELECT ano, COUNT(ano) AS `Total` FROM filmes GROUP BY ano ORDER BY Total DESC;
```

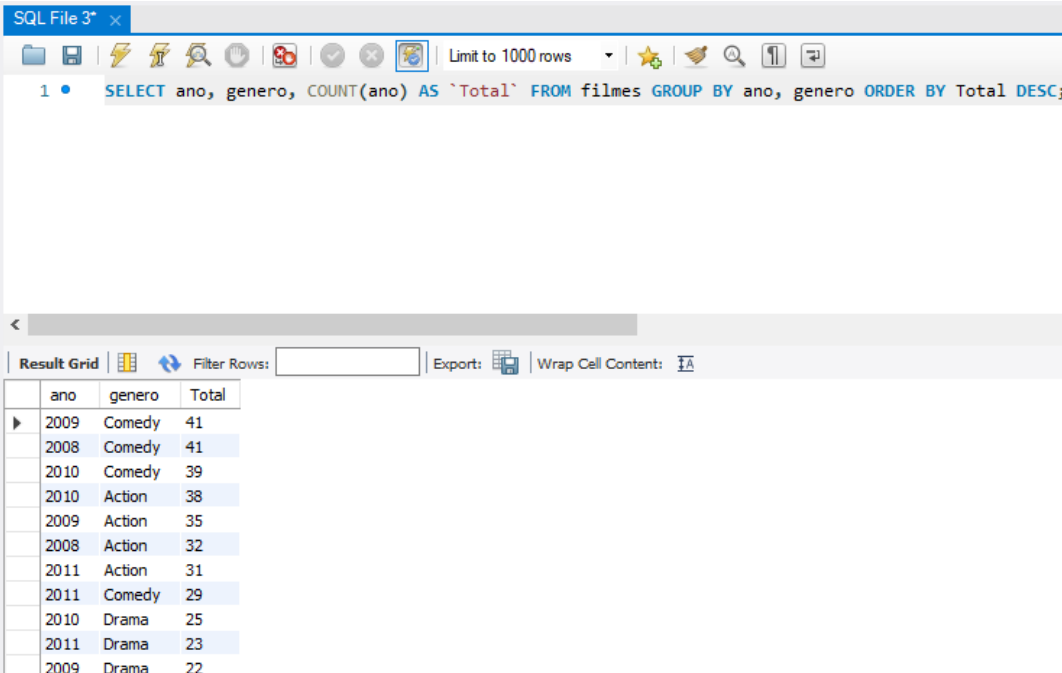


The screenshot shows a SQL editor window titled "SQL File 3\*" with a toolbar and a result grid. The query is: `SELECT ano, COUNT(ano) AS `Total` FROM filmes GROUP BY ano ORDER BY Total DESC;`. The result grid displays the following data:

ano	Total
2008	125
2011	123
2010	119
2009	116
2007	79

19. Qual gênero produziu mais filmes em um ano?

```
SELECT ano, genero, COUNT(ano) AS `Total` FROM filmes GROUP BY ano, genero ORDER BY Total DESC;
```

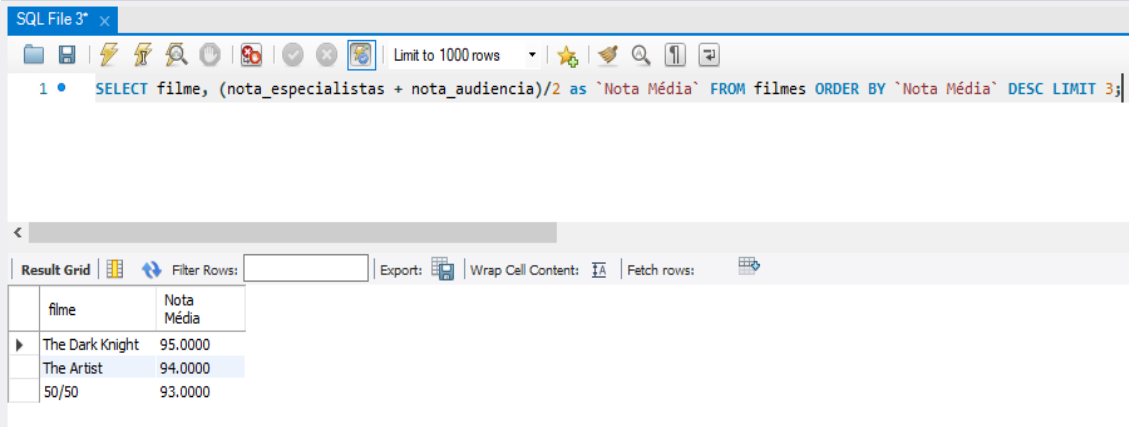


The screenshot shows a SQL editor window titled "SQL File 3\*" with a toolbar and a result grid. The query is: `SELECT ano, genero, COUNT(ano) AS `Total` FROM filmes GROUP BY ano, genero ORDER BY Total DESC;`. The result grid displays the following data:

ano	genero	Total
2009	Comedy	41
2008	Comedy	41
2010	Comedy	39
2010	Action	38
2009	Action	35
2008	Action	32
2011	Action	31
2011	Comedy	29
2010	Drama	25
2011	Drama	23
2009	Drama	22

20. Qual o filme mais amado pela audiência e pelos especialistas ao mesmo tempo?

```
SELECT filme, (nota_especialistas + nota_audiencia)/2 as `Nota Média` FROM  
filmes ORDER BY `Nota Média` DESC LIMIT 3;
```



The screenshot shows a SQL IDE window titled "SQL File 3". The query editor contains the SQL query: `SELECT filme, (nota_especialistas + nota_audiencia)/2 as `Nota Média` FROM filmes ORDER BY `Nota Média` DESC LIMIT 3;`. Below the query editor, the "Result Grid" tab is active, displaying the results of the query. The results are as follows:

filme	Nota Média
The Dark Knight	95.0000
The Artist	94.0000
50/50	93.0000