

Programação em SQL Iniciação

FEC.SQI.C.D.77

Horário, assiduidade e outras considerações

- Carga horária 60 horas (Presença obrigatória: 90% (54 horas))
- Toda a formação é síncrona sendo a sua presença obrigatória
- A assiduidade e pontualidade são registadas por unidades de 15 minutos
- A formação tem início às 19.00 e termina às 23.00 com intervalo de 30 minutos para descanso
- O horário está publicado no portal do formando e partilhado no calendário do TEAMS
- É da responsabilidade do formando garantir todas as condições técnicas para a participação na formação
- As faltas só precisam de ser justificadas por uma questão de cortesia com o formador.

Avaliação ao módulo

- Realização e entrega dentro dos prazos de 75% das tarefas propostas (40% da nota final)
- Ficha de trabalho final (50% da nota final com nota mínima de 8 valores em 20)
- Autoavaliação (10% da nota final)



Objetivos da formação

- Criar estruturas de Bases de dados em SQL;
 - Tabelas
 - VIEWS
 - Procedimentos
 - Triggers
- Manipular dados em tabelas
 - Inserir
 - Alterar
 - Atualizar
 - Apagar
- Consultar dados armazenados dentro da base de dados usando consultas simples, subconsultas
- Utilizar junções para seleção



O que é o SQL?

- SQL significa Structured Query Language;
- É uma linguagem padrão ANSI desde 1986 - (American National Standards Institute);
- O SQL é utilizado em Sistemas de Gestão de Base de Dados:
 - MySQL
 - SQL Server
 - MS Access
 - Oracle
 - PostgreSQL
 - ... outros



O que se pode fazer com SQL?

- Criar **bases de dados** (BD);
- Criar **tabelas** numa BD;
- **Inserir** registos (dados) numa BD;
- **Recuperar** (obter) dados de uma BD;
- **Atualizar** dados de uma BD;
- **Apagar** dados de uma BD;
- Criar **procedimentos** dados numa BD;
- Criar **views** numa BD;
- Definir **permissões** em tabelas, procedimentos e vistas numa BD;



A importância do SQL

- O SQL desempenha um papel fundamental na área de gestão de bases de dados e tem uma importância significativa por várias razões:
 - Linguagem padronizada
 - Gestão de bases de dados
 - Consultas complexas
 - Integração com outras linguagens e ferramentas
 - Segurança e controle de acesso
 - Escalabilidade e desempenho



A importância do SQL – **Linguagem padronizada**

- O SQL é uma linguagem padronizada e amplamente adotada para a gestão de bases de dados relacionais. Por ser padrão permite que os programadores e profissionais de bases de dados utilizem as mesmas habilidades e conhecimentos em diferentes sistemas de gestão de bases de dados, facilitando a portabilidade e a interoperabilidade.

A importância do SQL – **Gestão de dados**

- O SQL permite que os utilizadores realizem diversas operações em bases de dados, como criar, modificar e excluir tabelas, definir restrições de integridade, inserir, atualizar e excluir dados, bem como consultar e recuperar informações.
- O SQL fornece uma ampla gama de comandos para manipular dados e gerir a estrutura dos bases de dados de forma eficiente.



A importância do SQL – **Consultas complexas**

- O SQL oferece recursos poderosos para consultar bases de dados. Os utilizadores podem escrever consultas complexas para recuperar dados específicos com base em critérios definidos, combinar dados de várias tabelas usando junções, realizar operações de agregação, filtrar e classificar dados, entre outros. Essa capacidade de consulta sofisticada permite que as organizações extraiam informações valiosas de grandes volumes de dados.



A importância do SQL – **Integração com outras linguagens e ferramentas**

- O SQL pode ser integrado com várias linguagens de programação e ferramentas de desenvolvimento. Isso permite que os programadores acessem e manipulem bases de dados por meio de aplicativos e sistemas externos.
- Além disso, muitos sistemas de gestão de bases de dados fornecem interfaces e drivers que permitem a interação do SQL com linguagens como Java, Python, C#, entre outras.

A importância do SQL – **Segurança e controlo de acesso**

- O SQL possui recursos embutidos para garantir a segurança dos dados armazenados.
- Permite definir permissões e restrições de acesso para diferentes utilizadores e grupos, controlando quem pode executar determinadas operações numa base de dados. Isso ajuda a proteger os dados sensíveis e garante a conformidade com políticas de segurança e privacidade.

A importância do SQL – **Escalabilidade e desempenho**

- O SQL é projetado para lidar com grandes volumes de dados e oferecer bom desempenho.
- Os sistemas de gestão de bases de dados otimizam automaticamente as consultas SQL para executá-las de maneira eficiente, usando índices, otimização de consultas e outras técnicas. Isso permite que as bases de dados cresçam e atendam a demandas de desempenho, tornando-se escaláveis e eficientes.



Apesar de o SQL ser um padrão...

- Apesar do SQL ser um padrão ANSI e ISO existem diferentes versões da linguagem.
- Apesar de tudo, a maioria dos comandos (SELECT, UPDATE, DELETE, INSERT, WHERE) são de certa forma semelhantes.
- Quase todos os SGBD (programas) têm as suas próprias extensões proprietárias.



Exemplo de Estrutura de uma tabela

identificador da base de dados

identificador da tabela

v	formacao	curso
id	:	int(11)
curso	:	varchar(100)
regime	:	varchar(100)

colunas ou atributos da tabela

tipo de dados de cada atributo



Exemplo de registo guardados numa tabela

registos da tabela

colunas ou atributos da tabela

id	curso	regime
1	CET Multimedia	Pos Laboral
2	CET Multimedia	Laboral
3	CET Redes	Pos Laboral
4	CET Redes	Laboral
5	CET Sistemas de Informacao	Labora
6	CET Sistemas de Informacao	Pos Laboral
7	CET de Automacao	Laboral
8	CET de Automacao	Pos Laboral



Tabelas numa base de dados

- Uma base de dados é composta por uma ou mais tabelas;
- Cada tabela tem o seu nome ou identificador;
- As tabelas guardam dados;
- Os dados, dependendo da sua natureza, agrupam-se em tipos.

formacao formando	
id : int(11)	
nome : varchar(100)	
apelido : varchar(100)	
genero : varchar(100)	
cidade : varchar(100)	
dataNascimento : date	
situacaoEmprego : varchar(100)	
estadoCivil : varchar(100)	
habilitacao : char(20)	
telemovel : char(9)	

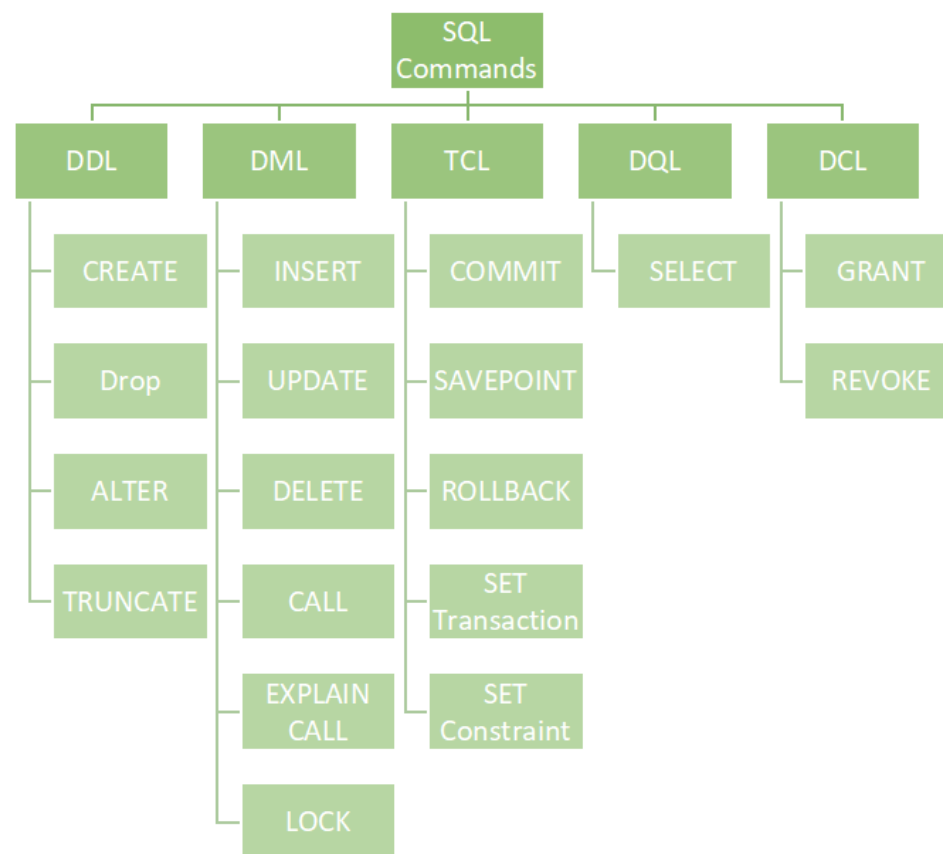
formacao curso	
id : int(11)	
curso : varchar(100)	
regime : varchar(100)	

formacao inscricao	
id : int(11)	
# idCurso : int(11)	
# idFormando : int(11)	

SQL – Comandos mais usados (alguns)

- SELECT - extrai dados;
- UPDATE - atualiza dados;
- DELETE – elimina dados;
- INSERT INTO – insere registos ou dados;
- CREATE DATABASE – cria uma base e dados;
- ALTER DATABASE – altera uma base de dados;
- CREATE TABLE – cria uma nova tabela;
- ALTER TABLE – modifica a estrutura de uma tabela;
- DROP TABLE – remove uma tabela;
- CREATE INDEX – cria um índice;
- DROP INDEX – remove o índice;

SQL – Comandos mais usados (terminologia)





SQL – **DDL** (Data Definition Language)

- DDL é a sigla para "Data Definition Language" ou "Linguagem de Definição de Dados" em português.
- Trata-se de um conjunto de comandos SQL utilizados para definir e gerir a estrutura de uma base de dados.
- Os comandos de DDL são responsáveis por criar, alterar e excluir objetos da base de dados, como tabelas, índices, views e restrições.

SQL – **DDL** (Data Definition Language)

1. **CREATE:** Utilizado para criar objetos do bases de dados, como tabelas, índices, visões, procedimentos armazenados, entre outros.
2. **ALTER:** Permite modificar a estrutura de objetos já existentes na base de dados. Por exemplo, é possível adicionar ou remover colunas de uma tabela, modificar o tipo de dado de uma coluna, ou renomear um objeto.
3. **DROP:** Utilizado para excluir objetos da base de dados, como tabelas, visões ou índices.
4. **TRUNCATE:** Remove todos os dados de uma tabela, mantendo a estrutura da tabela intacta.
5. **RENAME:** Permite renomear objetos do banco de dados, como tabelas, colunas ou views.
6. **GRANT e REVOKE:** Utilizados para conceder ou revogar permissões de acesso a objetos da base de dados.

SQL – **DML** (Data Manipulation Language)

- DML é a sigla para "Data Manipulation Language" ou "Linguagem de Manipulação de Dados" em português.
- Essa parte do SQL é responsável por executar operações de manipulação de dados numa base de dados.
- Os comandos DML permitem inserir, atualizar, excluir e consultar os dados armazenados em tabelas.



Ferramenta de texto para escrever SQL

- Bloco de notas
- Brackets
- Visual Code -- intelsense
- Noted pad++
- ...

Software: SGBD – XAMPP



The screenshot shows the XAMPP website with the following content:

- Navigation Bar:** Apache Friends, Download, Hosting, Community, About, Search bar, and language selector (EN).
- Header:** XAMPP Apache + MariaDB + PHP + Perl
- What is XAMPP?:**

XAMPP is the most popular PHP development environment

XAMPP is a completely free, easy to install Apache distribution containing MariaDB, PHP, and Perl. The XAMPP open source package has been set up to be incredibly easy to install and to use.
- Download Section:**
 - Download:** Click here for other versions
 - XAMPP for Windows:** 8.2.4 (PHP 8.2.4)
 - XAMPP for Linux:** 8.2.4 (PHP 8.2.4)
 - XAMPP for OS X:** 8.2.4 (PHP 8.2.4)

Software: SGBD – USBWEBSERVER



The screenshot shows the SourceForge project page for USBWebserver v10. The browser address bar displays 'sourceforge.net/projects/usbwebserver/'. The SourceForge logo is at the top left, followed by navigation tabs: 'Open Source Software', 'Business Software', and 'Resources'. Below these, the breadcrumb 'Home / Browse Open Source / USBWebserver v10' is visible. The main section features a server icon, the title 'USBWebserver v10', and the text 'Updated with the latest stable versions of PHP, Apache and PhpMyAdmin' and 'Brought to you by: tcpoverhttp'. It includes a green 'Download' button with the SourceForge logo, a green shield icon, and buttons for 'Get Updates' and 'Share This'. A 'Downloads: 245 This Week' badge is also present. Below this is a tabbed interface with 'Summary', 'Files', 'Reviews', and 'Support'. The 'Summary' tab is active, showing the text 'USBWebserver v10 Updated!' followed by a list of dependencies: '* Php 8.1.7', '* Apache 2.4.54', '* PhpMyAdmin 5.2.0', and '* MySQL 5.7.37'.

← → ↻ 🔒 sourceforge.net/projects/usbwebserver/

SOURCEFORGE

Open Source Software Business Software Resources

Home / Browse Open Source / USBWebserver v10

 **USBWebserver v10**
Updated with the latest stable versions of PHP, Apache and PhpMyAdmin
Brought to you by: [tcpoverhttp](#)

[Add a Review](#) **Downloads: 245 This Week**

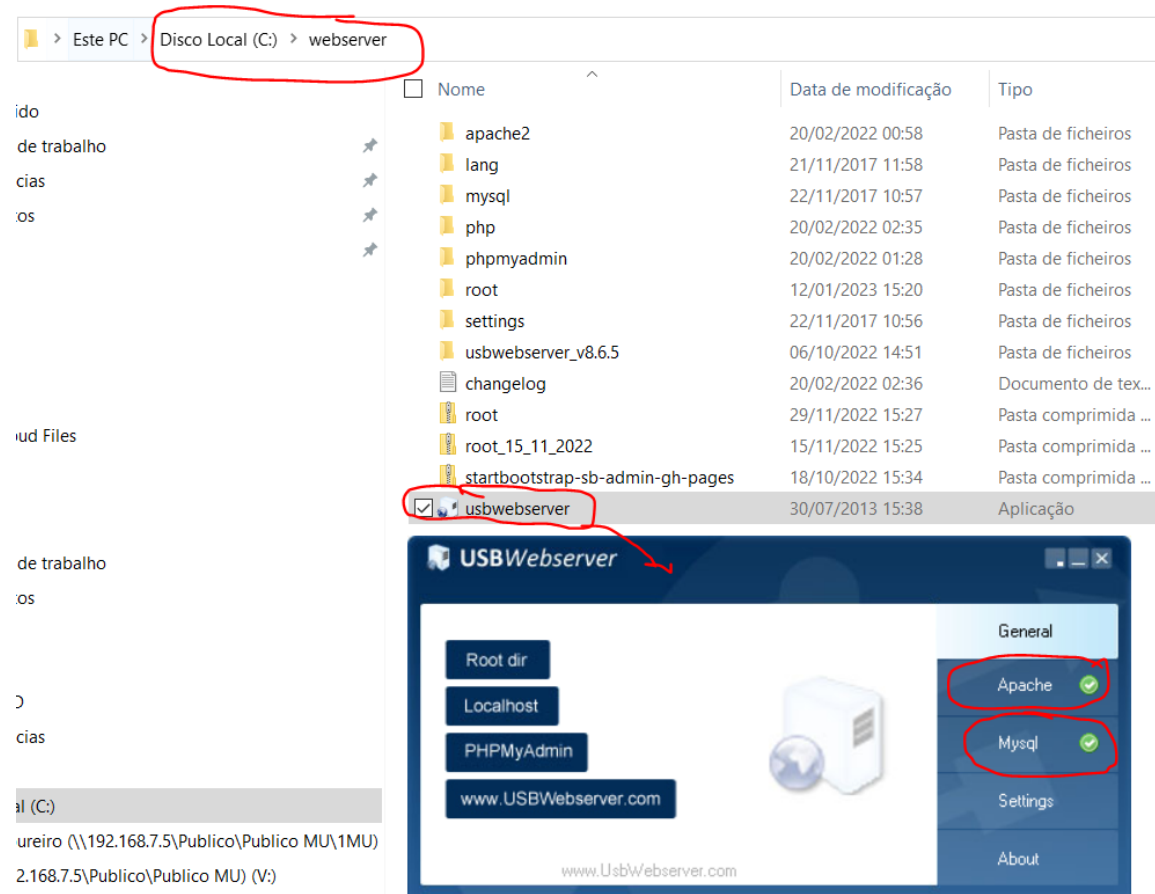
 **Download**  [Get Updates](#) [Share This](#)

[Summary](#) Files Reviews Support

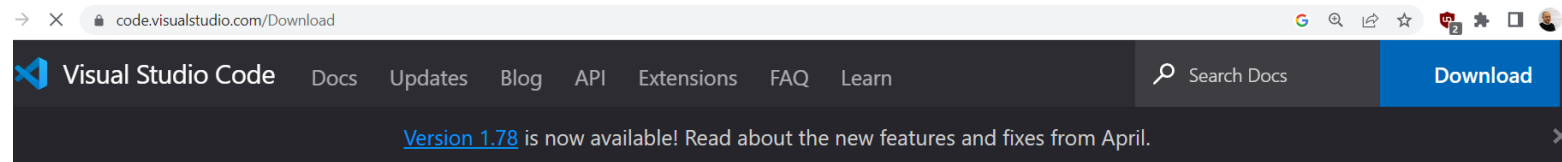
USBWebserver v10 Updated!

- * Php 8.1.7
- * Apache 2.4.54
- * PhpMyAdmin 5.2.0
- * MySQL 5.7.37

Software: SGBD – USBWEBSERVER



Software: Editor de texto – VS Code



Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



Windows

Windows 8, 10, 11

User Installer [x64](#) [x86](#) [Arm64](#)
System Installer [x64](#) [x86](#) [Arm64](#)

.deb

Debian, Ubuntu

.deb [x64](#) [Arm32](#) [Arm64](#)
.rpm [x64](#) [Arm32](#) [Arm64](#)

.rpm

Red Hat, Fedora, SUSE

Mac

macOS 10.11+

.zip [Intel chip](#) [Apple silicon](#) [Universal](#)



CINEL

Centro de Formação Profissional da Indústria Electrónica, Energia, Telecomunicações e Tecnologias da Informação

Software: Editor de texto – Instalação da extensão MYSQL

The screenshot displays the Visual Studio Code interface with the Extensions Marketplace open. The search bar contains 'mysql'. The left sidebar lists several MySQL-related extensions, with the top one, 'MySQL' by Jun Han, highlighted. The main panel shows the details for this extension, which is version v0.4.1 and has 1,339,215 downloads and a 4-star rating. It is currently installed globally, with 'Disable' and 'Uninstall' buttons visible. Below the extension details, there are tabs for 'DETAILS', 'FEATURE CONTRIBUTIONS', 'CHANGELOG', and 'RUNTIME STATUS'. The 'FEATURES' section lists several capabilities: Manage MySQL Connections (support SSL connection), List MySQL Servers, List MySQL Databases, List MySQL Tables, List MySQL Columns, and Run MySQL Query.

EXTENSIONS: MARKETPLACE

mysql

MySQL 170ms
MySQL management tool
Jun Han

MySQL 916K ★ 4
Database manager for MySQL/MariaD...
Weijan Chen **Install**

SQLTools MySQL/MariaDB 634K ★ 4
SQLTools MySQL/MariaDB
Matheus Teixeira **Install**

MySQL Syntax 470K ★ 3
MySQL syntax highlighting support
Jake Bathman **Install**

mysql-inline-decorator 118K ★ 3
Add color coding to inline MYSQL stri...
odubuc **Install**

MySQL Autocomplete 35K ★ 5
MySQL Syntax Autocomplete for Visual...
nspinozacr **Install**

MySQL Statement Scratc... 42K ★ 5
Easy mysql statement running with si...
Jared Black **Install**

Extension: MySQL v0.4.1
Jun Han | 1,339,215 | ★★★★★ (36)
MySQL management tool
Disable **Uninstall**
This extension is enabled globally.

MySQL
Build Status

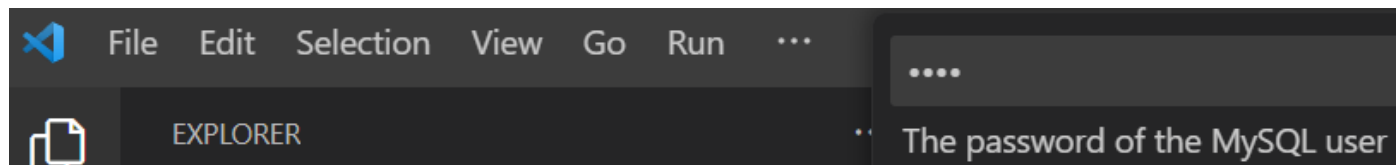
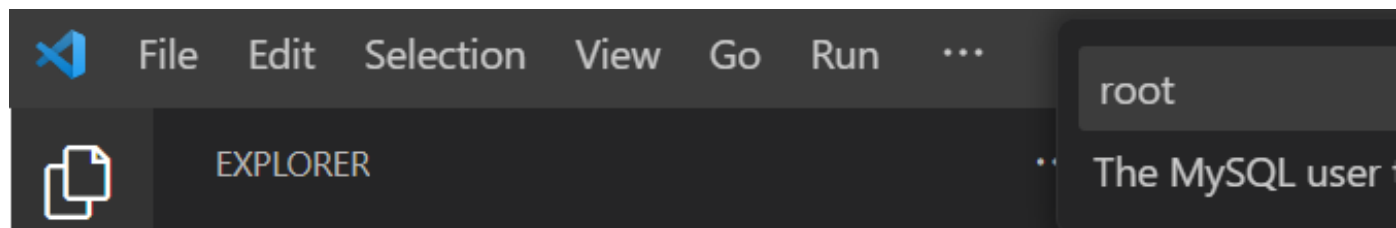
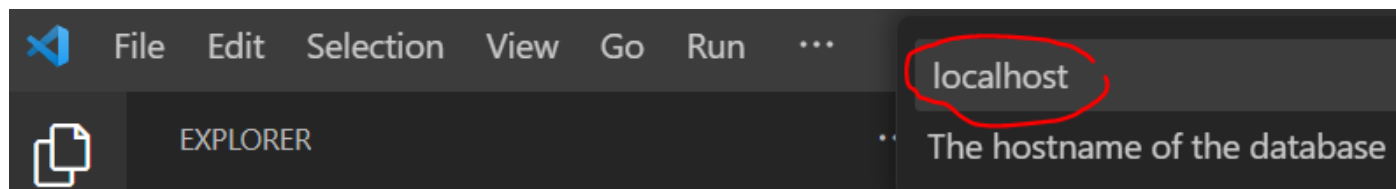
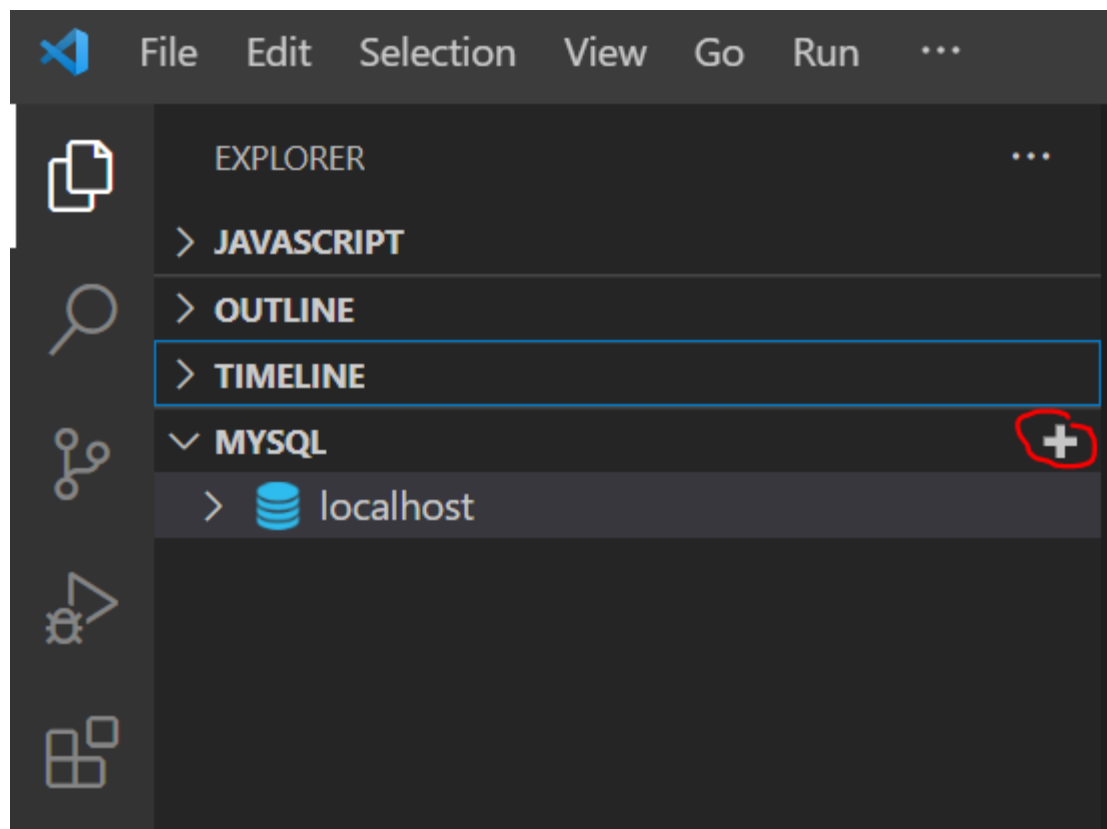
MySQL management tool

Features

- Manage MySQL Connections (support SSL connection)
- List MySQL Servers
- List MySQL Databases
- List MySQL Tables
- List MySQL Columns
- Run MySQL Query



Software: Editor de texto – Configuração da extensão MYSQL





SELECT – Hello World! – (A maldição do Olá Mundo)

```
Database formacao; Untitled-2 • introducao.sql • ... ts Result
C: > Users > mloureiro > Documents > sql > introducao.sql
1 SELECT 'Hello World!'
2
```

```
mloureiro > Documents > sql > introducao.sql
SELECT 'Hello World!' AS 'The curse'
```



SELECT – Exemplos com operações matemáticas

```
so; Untitled-1 • drop database formacao; Untitled-2 • introducao.sql • ... Results
C: > Users > mloureiro > Documents > sql > introducao.sql
1 SELECT 2 AS n1;
```

n1
2


```
drop database formacao; Untitled-2 • introducao.sql • ... Results
mloureiro > Documents > sql > introducao.sql
SELECT 2 AS n1, 3 AS n2;
```

n1	n2
2	3

```
mloureiro > Documents > sql > introducao.sql
SELECT 2 AS n1, 3 AS n2, 2+3 AS 'n1+n2';
```

n1	n2	n1+n2
2	3	5

SELECT – Exemplo com datas

> mloureiro > Documents > sql >  introducao.sql

```
SELECT NOW() AS 'data atual';
```

data atual

Tue May 16 2023 16:25:10 GMT+0100 (Western European Summer Time)

```
SELECT NOW() AS 'data atual';
```

data atual

Tue May 16 2023 16:25:10 GMT+0100 (Western European Summer Time)



SELECT – Exemplo com variáveis

```
> mloureiro > Documents > sqli > introducao.sql  
SET @n1 = 5;  
SELECT @n1 + 10 AS '@n1 + 10';
```

@n1 + 10

15



Exercício 0 - **SELECT**

n1	n2	$n1 + n2$	$n1 * n2$	$n1 - n2$	$n1 / n2$	$n1 \% n2$
7	4	11	28	3	1.75	3



Estruturas: DATABASE

The screenshot shows a MySQL IDE interface. On the left, the 'EXPLORER' panel displays a tree view of the database structure. Under the 'MYSQL' section, 'localhost' is expanded, showing several databases: 'information_sc...', 'formacao', 'mysql', and 'performance_s...'. The 'formacao' database is highlighted with a red underline. The main editor window shows a file named 'aula_1.sql' with the following SQL code:

```
C: > Users > mloureiro > Documents > sql > aula_1.sql  
1 CREATE DATABASE IF NOT EXISTS formacao;  
2
```

The SQL code is highlighted with a blue background, and the word 'formacao' is underlined with a red line.



Estruturas: DATABASE

The screenshot shows a code editor with a dark theme. On the left is an 'EXPLORER' sidebar with a tree view containing 'JAVASCRIPT', 'OUTLINE', 'TIMELINE', and 'MYSQL'. Under 'MYSQL', there is a 'localhost' folder containing 'information_sc...', 'mysql', and 'performance_s...'. The main editor area has a tab titled 'aula_1.sql'. Below the tab is the file path 'C: > Users > mloureiro > Documents > sql > aula_1.sql'. The editor contains two lines of SQL code: '1 CREATE DATABASE IF NOT EXISTS formacao;' and '2 DROP DATABASE IF EXISTS formacao;'. The second line is highlighted with a red oval.

```
EXPLORER ... aula_1.sql
> JAVASCRIPT
> OUTLINE
> TIMELINE
✓ MYSQL +
  ✓ localhost
    > information_sc...
    > mysql
    > performance_s...

C: > Users > mloureiro > Documents > sql > aula_1.sql
1 CREATE DATABASE IF NOT EXISTS formacao;
2 DROP DATABASE IF EXISTS formacao;
```



Estruturas: DATABASE

The screenshot shows a SQL IDE interface. On the left is an 'EXPLORER' pane with a tree view containing 'JAVASCRIPT', 'OUTLINE', 'TIMELINE', and 'MYSQL'. Under 'MYSQL', 'localhost' is expanded, showing databases: 'information_sc...', 'formacao' (underlined), 'mysql', and 'performance_s...'. The main editor pane shows a file named 'aula_1.sql' with the following SQL code:

```
C: > Users > mloureiro > Documents > sql > aula_1.sql  
1 DROP DATABASE IF EXISTS formacao;  
2 CREATE DATABASE IF NOT EXISTS formacao;
```

The two lines of SQL code are highlighted with a red oval.



Estruturas: Tabela

```
CREATE TABLE IF NOT EXISTS curso(  
  id INT  
);
```

```
DROP TABLE IF EXISTS curso;
```



Estruturas: Tabela

```
DROP TABLE IF EXISTS curso;
```

```
CREATE TABLE curso(  
    idCurso INT  
);
```



Estruturas: Tabela

```
DESCRIBE curso;
```

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	YES		null	



Estruturas: Tabela - **CREATE**

```
DROP TABLE IF EXISTS curso;
```

```
CREATE TABLE curso(  
    idCurso INT  
);
```

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	YES		null	



Estruturas: Tabela - **ALTER**

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	YES		null	
designacao	text	YES		null	



Estruturas: Tabela - **ALTER**

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	YES		null	
designacao	text	YES		null	

```
ALTER TABLE curso ADD IF NOT EXISTS designacao TEXT;
```

```
DESCRIBE curso;
```



Estruturas: Tabela – **ALTER ... ADD**

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	YES		null	
regime	text	YES		null	
designacao	text	YES		null	

```
ALTER TABLE curso ADD IF NOT EXISTS regime TEXT AFTER idCurso;
```

```
DESCRIBE curso;
```



Estruturas: Tabela – **ALTER ... MODIFY**

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	YES		null	
<u>designacao</u>	text	YES		null	
regime	text	YES		null	

```
ALTER TABLE curso MODIFY designacao TEXT AFTER idCurso;
```

```
DESCRIBE curso;
```



Estruturas: Tabela – **ALTER ... MODIFY**

Field	Type	Null	Key	Default	Extra
regime	text	YES		null	
idCurso	int(11)	YES		null	
designacao	text	YES		null	

```
ALTER TABLE curso MODIFY regime TEXT FIRST;
```

```
DESCRIBE curso;
```



Estruturas: Tabela – **ALTER ... MODIFY**

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	YES		null	
designacao	text	YES		null	
regime	text	YES		null	

```
ALTER TABLE curso MODIFY regime TEXT AFTER designacao;
```

```
DESCRIBE curso;
```



Estruturas: Tabela – **ALTER ... CHANGE**

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	YES		null	
designacaoCurso	text	YES		null	
regime	text	YES		null	

```
ALTER TABLE curso CHANGE designacao designacaoCurso TEXT;
```

```
DESCRIBE curso;
```




Estruturas: Tabela – **ALTER ... CHANGE**

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	YES		null	
designacao	text	YES		null	
regime	text	YES		null	

```
ALTER TABLE curso CHANGE designacaoCurso designacao TEXT;
```

```
DESCRIBE curso;
```



Estruturas: Tabela – **ALTER ... ADD**

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	YES		null	
designacao	text	YES		null	
regime	text	YES		null	
estado	int(1)	YES		1	

```
ALTER TABLE curso ADD estado INT(1) DEFAULT 1;
```

```
DESCRIBE curso;
```



Estruturas: Tabela – **ALTER ... DROP**

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	YES		null	
designacao	text	YES		null	
regime	text	YES		null	

```
ALTER TABLE curso DROP estado;
```

```
DESCRIBE curso;
```

SQL Constraints (Limitações ou restrições)

- Especificam regras na manipulação de dados da tabela
- Podem ser ao nível da tabela ou da coluna

NOT NULL – Garante que o campo na tabela não recebe um valor nulo;

UNIQUE – Garante que o valor não é repetido na tabela;

PRIMARY KEY – Combinação entre o NOT NULL e o UNIQUE;

FOREIGN KEY – Primary KEY noutra tabela; **DEFAULT** – Assegura um valor predeterminado caso a coluna não receba outro valor;

INDEX – Para fazer ligação entre colunas da tabela.

CHECK – Assegura que um valor da coluna satisfaz uma dada condição; (Não suportada em MYSQL)



Estruturas: CONSTRAINTS (Restrições)

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	YES		null	
designacao	text	YES		null	
regime	text	NO		null	

```
ALTER TABLE curso CHANGE regime regime TEXT NOT NULL;
```

```
DESCRIBE curso;
```



Estruturas: CONSTRAINTS (Restrições)

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	YES		null	
<u>designacao</u>	text	YES	UNI	null	
regime	text	NO		null	

```
ALTER TABLE curso ADD UNIQUE (designacao);
```

```
DESCRIBE curso;
```



Estruturas: CONSTRAINTS (Restrições)

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	NO	PRI	null	
designacao	text	YES		null	
regime	text	YES		null	

```
ALTER TABLE curso ADD PRIMARY KEY(idCurso);
```

```
DESCRIBE curso;
```



Estruturas: CONSTRAINTS (AUTO_INCREMENT)

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	NO	PRI	null	auto_increment
designacao	text	YES	UNI	null	
regime	text	YES		null	

```
ALTER TABLE tb_curso CHANGE idCurso idCurso INT AUTO_INCREMENT;
```

```
describe tb_curso;
```




Estruturas: CONSTRAINTS (Restrições)

```
ALTER TABLE curso DROP INDEX designacao;
```

```
DESCRIBE curso;
```

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	NO	PRI	null	
designacao	text	YES		null	
regime	text	NO		null	



Estruturas: CONSTRAINTS (Restrições)

Voltamos a adicionar a restrição UNIQUE.

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	YES		null	
<u>designacao</u>	text	YES	UNI	null	
regime	text	NO		null	

```
ALTER TABLE curso ADD UNIQUE (designacao);
```

```
DESCRIBE curso;
```



Criação de estruturas em SQL – Tipos de dados

Um programador SQL deve decidir que tipo de dados serão armazenados dentro de cada coluna ao criar uma tabela.

O tipo de dados é uma diretriz para o SQL entender que tipo de dados é esperado dentro de cada coluna e também identifica como o SQL irá interagir com os dados armazenados.

No MySQL existem três tipos de dados principais: string, numérico e data e hora.

Criação de estruturas em SQL – Tipos de dados: String

TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT(size)	Holds a string with a maximum length of 65,535 bytes
BLOB(size)	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data

Criação de estruturas em SQL – Tipos de dados: String

ENUM(val1, val2, val3, ...)

A string object that can have only one value, chosen from a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them

SET(val1, val2, val3, ...)

A string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list

Criação de estruturas em SQL – Tipos de dados: Numérico

Data type	Description
BIT(<i>size</i>)	A bit-value type. The number of bits per value is specified in <i>size</i> . The <i>size</i> parameter can hold a value from 1 to 64. The default value for <i>size</i> is 1.
TINYINT(<i>size</i>)	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The <i>size</i> parameter specifies the maximum display width (which is 255)
BOOL	Zero is considered as false, nonzero values are considered as true.
BOOLEAN	Equal to BOOL
SMALLINT(<i>size</i>)	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The <i>size</i> parameter specifies the maximum display width (which is 255)



Criação de estruturas em SQL – Tipos de dados: Numérico

MEDIUMINT(*size*)

A medium integer. Signed range is from -8388608 to 8388607.
Unsigned range is from 0 to 16777215. The *size* parameter specifies the maximum display width (which is 255)

INT(*size*)

A medium integer. Signed range is from -2147483648 to 2147483647.
Unsigned range is from 0 to 4294967295. The *size* parameter specifies the maximum display width (which is 255)

INTEGER(*size*)

Equal to INT(*size*)

BIGINT(*size*)

A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The *size* parameter specifies the maximum display width (which is 255)

Criação de estruturas em SQL – Tipos de dados: Numérico

`FLOAT(size, d)`

A floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions

`FLOAT(p)`

A floating point number. MySQL uses the *p* value to determine whether to use FLOAT or DOUBLE for the resulting data type. If *p* is from 0 to 24, the data type becomes FLOAT(). If *p* is from 25 to 53, the data type becomes DOUBLE()

`DOUBLE(size, d)`

A normal-size floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter

`DOUBLE PRECISION(size, d)`



Criação de estruturas em SQL – Tipos de dados: Numérico

DECIMAL(*size*, *d*)

An exact fixed-point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. The maximum number for *size* is 65. The maximum number for *d* is 30. The default value for *size* is 10. The default value for *d* is 0.

DEC(*size*, *d*)

Equal to DECIMAL(*size*,*d*)

Criação de estruturas em SQL – Tipos de dados: Data e hora

Data type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME(<i>fsp</i>)	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time

Criação de estruturas em SQL – Tipos de dados: Data e hora

TIMESTAMP(<i>fsp</i>)	A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column definition
TIME(<i>fsp</i>)	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format.

Criação de estruturas em SQL – Tipos de dados: String

Data type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The <i>size</i> parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The <i>size</i> parameter specifies the maximum column length in characters - can be from 0 to 65535
BINARY(size)	Equal to CHAR(), but stores binary byte strings. The <i>size</i> parameter specifies the column length in bytes. Default is 1
VARBINARY(size)	Equal to VARCHAR(), but stores binary byte strings. The <i>size</i> parameter specifies the maximum column length in bytes.
TINYBLOB	For BLOBs (Binary Large Objects). Max length: 255 bytes



Exercício 1 - Criação da tabela **tb_genero**

Field	Type	Null	Key	Default	Extra
idGenero	int(11)	NO	PRI	null	auto_increment
genero	text	NO	UNI	null	

```
CREATE TABLE tb_genero(  
  idGenero INT AUTO-INCREMENT PRIMARY KEY,  
  genero TEXT UNIQUE NOT NULL  
);
```



Exercício 2 - Criação da tabela – **tb_cidade**

Field	Type	Null	Key	Default	Extra
idCidade	int(11)	NO	PRI	null	auto_increment
cidade	text	NO	UNI	null	

```
CREATE TABLE tb_cidade(  
  idCidade INT AUTO_INCREMENT PRIMARY KEY,  
  cidade TEXT UNIQUE NOT NULL  
);
```



Exercício 3 - Criação da tabela – **tb_emprego**

Field	Type	Null	Key	Default	Extra
idEmprego	int(11)	NO	PRI	null	auto_increment
emprego	text	NO	UNI	null	

```
CREATE TABLE tb_emprego(  
  idEmprego INT AUTO_INCREMENT PRIMARY KEY,  
  emprego TEXT UNIQUE NOT NULL  
);
```



Exercício 4 - Criação da tabela – **tb_estadoCivil**

Field	Type	Null	Key	Default	Extra
idEstadoCivil	int(11)	NO	PRI	null	auto_increment
estadoCivil	text	NO	UNI	null	

```
CREATE TABLE tb_estadoCivil(  
  idEstadoCivil INT AUTO_INCREMENT PRIMARY KEY,  
  estadoCivil TEXT UNIQUE NOT NULL  
);
```




Exercício 5 - Criação da tabela – **tb_habilitacao**

Field	Type	Null	Key	Default	Extra
idHabilitacao	int(11)	NO	PRI	null	auto_increment
habilitacao	text	NO	UNI	null	

```
create table tb_habilitacao(  
  idHabilitacao INT AUTO_INCREMENT PRIMARY KEY,  
  habilitacao TEXT UNIQUE NOT NULL  
);
```



Exercício 6 - Criação da tabela – **tb_formando**

Field	Type	Null	Key	Default	Extra
idFormando	int(11)	NO	PRI	null	auto_increment
nome	text	NO		null	
apelido	text	NO		null	
dataNascimento	date	YES		null	
telemovel	text	YES		null	

```
create table tb_formando(  
  idFormando INT AUTO_INCREMENT PRIMARY KEY,  
  nome TEXT NOT NULL,  
  apelido TEXT NOT NULL,  
  dataNascimento DATE,  
  telemovel TEXT  
);
```



Criação de estruturas em SQL – **RENAME ... TO**

```
RENAME TABLE curso TO tb_curso;
```

```
DESCRIBE tb_curso;
```

Field	Type	Null	Key	Default	Extra
idCurso	int(11)	NO	PRI	null	
designacao	text	YES		null	
regime	text	NO		null	

```
DESCRIBE tb_curso;
+-----+
| idCurso | int(11) | NO | PRI | null |  |
+-----+
| designacao | text | YES |  | null |  |
+-----+
| regime | text | NO |  | null |  |
+-----+
```



Criação de estruturas em SQL – **SHOW TABLES**

```
SHOW TABLES
```

```
Tables_in_formacao
```

```
tb_cidade
```

```
tb_curso
```

```
tb_emprego
```

```
tb_estadocivil
```

```
tb_formando
```

```
tb_genero
```

```
tb_habilitacao
```



Inserção de registos numa tabela – **tb_curso**

idCurso	designacao	regime
1	CET de Multimédia	Laboral

```
INSERT INTO tb_curso (idCurso, designacao, regime)  
VALUES (NULL, 'CET de Multimédia', 'Laboral');
```

```
SELECT * FROM tb_curso;
```



Inserção de registos numa tabela – **tb_curso**

idCurso	designacao	regime
1	CET de Multimédia	Laboral
2	CET de Redes	Laboral

```
INSERT INTO tb_curso VALUES (NULL, 'CET de Redes', 'Laboral');
```

```
SELECT * FROM tb_curso;
```



Inserção de registos numa tabela – **tb_curso**

idCurso	designacao	regime
1	CET de Multimédia	Laboral
2	CET de SI	Laboral
3	EFA de Multimédia	Laboral
4	CET de Redes	Laboral
5	EFA de SI	Laboral

```
USE formacao;  
TRUNCATE TABLE tb_curso;  
INSERT INTO tb_curso (designacao, regime)  
VALUES ('CET de Multimédia', 'Laboral'),  
('CET de SI', 'Laboral'),  
('EFA de Multimédia', 'Laboral'),  
('CET de Redes', 'Laboral'),  
('EFA de SI', 'Laboral');  
  
select * from tb_curso;
```



Exercício 7 - Inserção de registos na tabela **tb_genero**

Field	Type	Null	Key	Default	Extra
idGenero	int(11)	NO	PRI	null	auto_increment
genero	text	YES		null	

idGenero	genero
1	Masculino
2	Feminino
3	Não divulgado

```
INSERT INTO tb_genero (genero) VALUES  
('Masculino'), ('Feminino'), ('Não divulgado');  
  
SELECT * FROM tb_genero;
```




Exercício 8 - Inserção de registos na tabela **tb_cidade**

Field	Type	Null	Key	Default	Extra
idCidade	int(11)	NO	PRI	null	auto_increment
cidade	text	YES		null	

idCidade	cidade
1	Coimbra
2	Faro
3	Setúbal
4	Aveiro
5	Porto

```
INSERT INTO tb_cidade (cidade) VALUES  
('Coimbra'), ('Faro'), ('Setúbal'), ('Aveiro'), ('Porto');  
  
SELECT * FROM tb_cidade;
```



Exercício 9 - Inserção de registos na tabela **tb_emprego**

Field	Type	Null	Key	Default	Extra
idEmprego	int(11)	NO	PRI	null	auto_increment
emprego	text	YES		null	

idEmprego	emprego
1	Empregado
2	Desempregado
3	Estudante
4	Reformado

```
INSERT INTO tb_emprego (emprego) VALUES  
('Empregado'), ('Desempregado'), ('Estudante'), ('Reformado');  
  
SELECT * FROM tb_emprego;
```



Exercício 10 - Inserção de registos na tabela **tb_estadoCivil**

Field	Type	Null	Key	Default	Extra
idEstadoCivil	int(11)	NO	PRI	null	auto_increment
estadoCivil	text	YES		null	

idEstadoCivil	estadoCivil
1	Solteiro
2	Casado
3	Comunhão
4	Divorciado
5	Viúvo

```
INSERT INTO tb_estadoCivil (estadoCivil) VALUES  
( 'Solteiro' ), ( 'Casado' ), ( 'Comunhão' ), ( 'Divorciado' ), ( 'Viúvo' );  
  
SELECT * FROM tb_estadoCivil;
```



Exercício 11 - Inserção de registos na tabela **tb_habilitacao**

Field	Type	Null	Key	Default	Extra
idHabilitacao	int(11)	NO	PRI	null	auto_increment
habilitacao	text	YES		null	

idHabilitacao	habilitacao
1	Básico
2	Secundário
3	Superior

```
INSERT INTO tb_habilitacao (habilitacao) VALUES  
('Básico'), ('Secundário'), ('Superior');  
  
SELECT * FROM tb_habilitacao;
```



UPDATE -

idHabilitacao	habilitacao
1	Básico
2	Secundário
3	Superior
4	Mestrado

idHabilitacao	habilitacao
1	Básico
2	Secundário
3	Superior
4	Mestre

```
USE formacao;  
UPDATE tb_habilitacao  
SET habilitacao = 'Mestre' WHERE habilitacao = 'Mestrado';  
  
SELECT * FROM tb_habilitacao;
```



UPDATE -

idCurso	designacao	regime
1	CET de Multimédia	Laboral
2	CET de SI	Laboral
3	EFA de Multimédia	Laboral
4	CET de Redes	Laboral
5	EFA de SI	Laboral

idCurso	designacao	regime
1	CET de Multimédia	LAB
2	CET de SI	LAB
3	EFA de Multimédia	LAB
4	CET de Redes	LAB
5	EFA de SI	LAB

```
USE formacao;  
UPDATE tb_curso  
SET regime = 'LAB' WHERE regime = 'laboral';  
SELECT * FROM tb_curso;
```



DELETE -

idHabilitacao	habilitacao
1	Básico
2	Secundário
3	Superior
4	Mestre

idHabilitacao	habilitacao
1	Básico
2	Secundário
3	Superior

```
USE formacao;  
DELETE FROM tb_habilitacao WHERE idHabilitacao = 4;  
SELECT * FROM tb_habilitacao;
```



DELETE -

idHabilitacao	habilitacao
1	Básico
2	Secundário
3	Superior
5	Doutor

idHabilitacao	habilitacao
1	Básico
2	Secundário
3	Superior

```
USE formacao;  
DELETE FROM tb_habilitacao WHERE idHabilitacao = 5;  
ALTER TABLE tb_habilitacao AUTO_INCREMENT = 4;  
SELECT * FROM tb_habilitacao;
```




SHOW -

Table	Create Table
tb_habilitacao	CREATE TABLE `tb_habilitacao` (`idHabilitacao` int(11) NOT NULL AUTO_INCREMENT, `habilitacao` text NOT NULL, PRIMARY KEY (`idHabilitacao`), UNIQUE KEY `habilitacao` (`habilitacao`) USING HASH) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4

```
USE formacao;  
SHOW CREATE TABLE tb_habilitacao;
```

E se houver um curso com 2 regimes ou mais?

idCurso	designacao	regime
1	CET de Multimédia	Laboral, Pós Laboral
2	CET de SI	Laboral
3	EFA de Multimédia	Laboral
4	CET de Redes	Laboral
5	EFA de SI	Laboral

```
UPDATE tb_curso  
SET regime = 'Laboral, Pós Laboral'  
WHERE idCurso=1;
```

idCurso	designacao	regime
1	CET de Multimédia	Laboral, Pós Laboral
2	CET de SI	Laboral
3	EFA de Multimédia	Laboral, Pós Laboral
4	CET de Redes	Laboral
5	EFA de SI	Laboral

```
UPDATE tb_curso  
SET regime = 'Laboral, Pós Laboral'  
WHERE idCurso=3;
```



CREATE TABLE

Field	Type	Null	Key	Default	Extra
idRegime	int(11)	NO	PRI	null	auto_increment
regime	text	NO	UNI	null	

```
USE formacao;
```

```
DROP TABLE IF EXISTS tb_regime;
```

```
CREATE TABLE tb_regime(  
    idRegime INT AUTO_INCREMENT PRIMARY KEY,  
    regime TEXT NOT NULL UNIQUE  
);
```

```
DESCRIBE tb_regime;
```



CREATE TABLE

idCurso	designacao
1	CET de Multimédia
2	CET de SI
3	EFA de Multimédia
4	CET de Redes
5	EFA de SI

```
USE formacao;  
ALTER TABLE tb_curso DROP regime;  
SELECT * FROM tb_curso;
```



CREATE TABLE

idCurso	designacao
1	CET de Multimédia
2	CET de SI
3	EFA de Multimédia
4	CET de Redes
5	EFA de SI

idRegime	regime
1	Laboral
2	Pós laboral

idCursoRegime	idCurso	idRegime
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	1	2
7	2	2
8	2	3

```
INSERT INTO curso_regime (idCurso, idRegime) VALUES  
(1,1),(2,1),(3,1),(4,1),(5,1),(1,2),(2,2),(2,3);
```



CREATE TABLE

idCurso	designacao
1	CET de Multimédia
2	CET de SI
3	EFA de Multimédia
4	CET de Redes
5	EFA de SI

idRegime	regime
1	Laboral
2	Pós laboral

idCurso	designacao	idRegime	regime
1	CET de Multimédia	1	Laboral
2	CET de SI	1	Laboral
3	EFA de Multimédia	1	Laboral
4	CET de Redes	1	Laboral
5	EFA de SI	1	Laboral
1	CET de Multimédia	2	Pós laboral
2	CET de SI	2	Pós laboral



CREATE TABLE

idCurso	designacao
1	CET de Multimédia
2	CET de SI
3	EFA de Multimédia
4	CET de Redes
5	EFA de SI

```
USE formacao;  
ALTER TABLE tb_curso DROP regime;  
SELECT * FROM tb_curso;
```

idRegime	regime
1	Laboral
2	Pós laboral

```
USE formacao;  
INSERT INTO tb_regime (regime)  
VALUES ('Laboral'), ('Pós laboral');  
SELECT * FROM tb_regime;
```



CREATE TABLE

idCurso	designacao
1	CET de Multimédia
2	CET de SI
3	EFA de Multimédia
4	CET de Redes
5	EFA de SI

idRegime	regime
1	Laboral
2	Pós laboral

idCursoRegime	idCurso	idRegime
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	1	2
7	2	2
8	2	3



CREATE TABLE

Field	Type	Null	Key	Default	Extra
idCursoRegime	int(11)	NO	PRI	null	auto_increment
idCurso	int(11)	NO		null	
idRegime	int(11)	NO		null	

```
INSERT INTO curso_regime (idCurso, idRegime) VALUES  
(1,1),(2,1),(3,1),(4,1),(5,1),(1,2),(2,2),(2,3);
```



CREATE TABLE

Field	Type	Null	Key	Default	Extra
idCursoRegime	int(11)	NO	PRI	null	auto_increment
idCurso	int(11)	NO		null	
idRegime	int(11)	NO		null	

```
INSERT INTO curso_regime (idCurso, idRegime) VALUES  
(1,1),(2,1),(3,1),(4,1),(5,1),(1,2),(2,2),(2,3);
```

idCursoRegime	idCurso	idRegime
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	1	2
7	2	2
8	2	3



CREATE TABLE

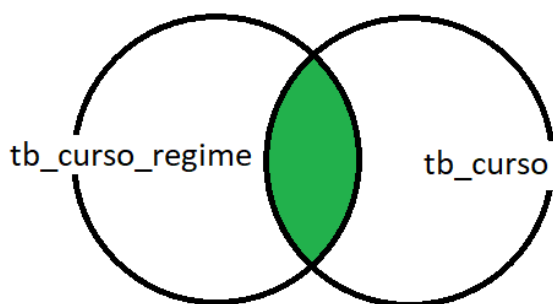
Como podemos ver agora a designação de um curso e o seu regime se os valores estão guardados em tabelas diferentes?

designacao	regime
CET de Multimédia	Laboral
CET de SI	Laboral
EFA de Multimédia	Laboral
CET de Redes	Laboral
EFA de SI	Laboral
CET de Multimédia	Pós laboral
CET de SI	Pós laboral



CREATE TABLE

Como



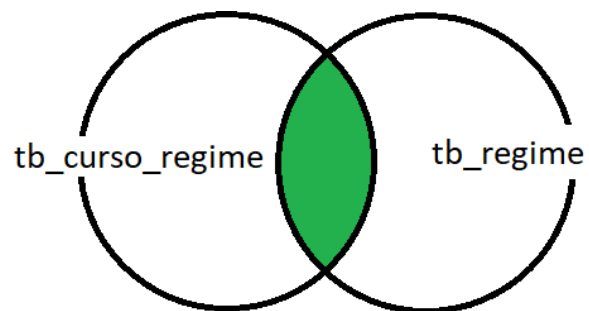
```
SELECT designacao FROM  
curso_regime JOIN tb_curso USING(idCurso);
```

designacao
CET de Multimédia
CET de SI
EFA de Multimédia
CET de Redes
EFA de SI
CET de Multimédia
CET de SI
CET de SI



CREATE TABLE

Como



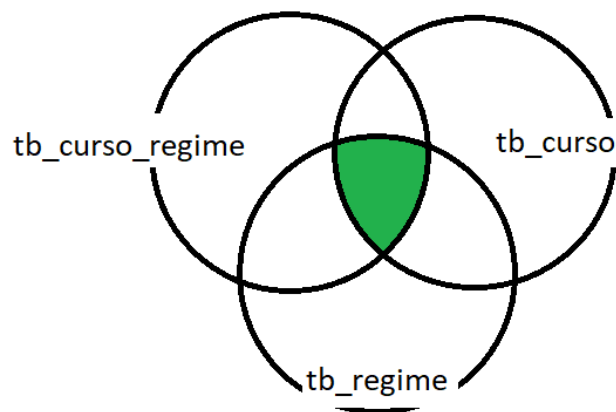
```
SELECT regime FROM  
curso_regime JOIN tb_regime USING (idRegime);
```

regime
Laboral
Laboral
Laboral
Laboral
Laboral
Pós laboral
Pós laboral



CREATE TABLE

As 3 tabelas agora



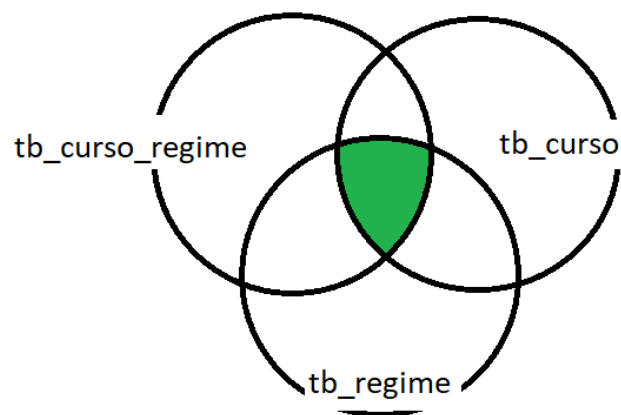
```
SELECT designacao, regime FROM  
curso_regime JOIN tb_curso USING(idCurso) JOIN  
tb_regime USING (idRegime);
```

designacao	regime
CET de Multimédia	Laboral
CET de SI	Laboral
EFA de Multimédia	Laboral
CET de Redes	Laboral
EFA de SI	Laboral
CET de Multimédia	Pós laboral
CET de SI	Pós laboral



CREATE TABLE

As 3 tabelas com todos as colunas intervenientes



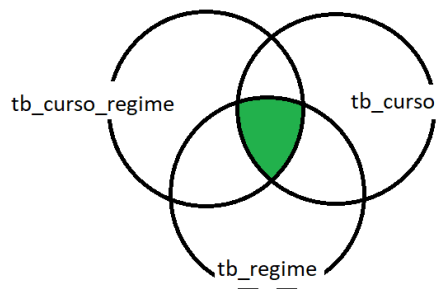
```
SELECT * FROM curso_regime  
JOIN tb_curso USING(idCurso)  
JOIN tb_regime USING (idRegime);
```

idRegime	idCurso	idCursoRegime	designacao	regime
1	1	1	CET de Multimédia	Laboral
1	2	2	CET de SI	Laboral
1	3	3	EFA de Multimédia	Laboral
1	4	4	CET de Redes	Laboral
1	5	5	EFA de SI	Laboral
2	1	6	CET de Multimédia	Pós laboral
2	2	7	CET de SI	Pós laboral



CREATE TABLE

Selecionando apenas 3 colunas



```
SELECT idCursoRegime, designacao, regime  
FROM curso_regime JOIN tb_curso USING(idCurso)  
JOIN tb_regime USING (idRegime);
```

idCursoRegime	designacao	regime
1	CET de Multimédia	Laboral
2	CET de SI	Laboral
3	EFA de Multimédia	Laboral
4	CET de Redes	Laboral
5	EFA de SI	Laboral
6	CET de Multimédia	Pós laboral
7	CET de SI	Pós laboral

Ordenação

Ordenação ascendente por designacao

idCursoRegime	designacao	regime
1	CET de Multimédia	Laboral
6	CET de Multimédia	Pós laboral
4	CET de Redes	Laboral
2	CET de SI	Laboral
7	CET de SI	Pós laboral
3	EFA de Multimédia	Laboral
5	EFA de SI	Laboral

```
SELECT idCursoRegime, designacao, regime  
FROM curso_regime JOIN tb_curso USING(idCurso)  
JOIN tb_regime USING (idRegime)  
ORDER BY designacao ASC
```

```
SELECT idCursoRegime, designacao, regime  
FROM curso_regime JOIN tb_curso USING(idCurso)  
JOIN tb_regime USING (idRegime)  
ORDER BY 2 ASC
```



Ordenação

Ordenação descendente por designação e ascendente por regime

idCursoRegime	designacao	regime
5	EFA de SI	Laboral
3	EFA de Multimédia	Laboral
2	CET de SI	Laboral
7	CET de SI	Pós laboral
4	CET de Redes	Laboral
1	CET de Multimédia	Laboral
6	CET de Multimédia	Pós laboral

```
SELECT idCursoRegime, designacao, regime
FROM curso_regime JOIN tb_curso USING(idCurso)
JOIN tb_regime USING (idRegime)
ORDER BY 2 DESC, 3 ASC
```

LIMIT

Os 3 primeiros

idCursoRegime	designacao	regime
1	CET de Multimédia	Laboral
6	CET de Multimédia	Pós laboral
4	CET de Redes	Laboral

```
SELECT idCursoRegime, designacao, regime
FROM curso_regime JOIN tb_curso USING(idCurso)
JOIN tb_regime USING (idRegime)
ORDER BY 2 -- Ascendente por defeito
LIMIT 3
```

idCursoRegime	designacao	regime
1	CET de Multimédia	Laboral
6	CET de Multimédia	Pós laboral
4	CET de Redes	Laboral
2	CET de SI	Laboral
7	CET de SI	Pós laboral
3	EFA de Multimédia	Laboral
5	EFA de SI	Laboral

LIMIT

Os 3 primeiros

idCursoRegime	designacao	regime
1	CET de Multimédia	Laboral
6	CET de Multimédia	Pós laboral
4	CET de Redes	Laboral

```
SELECT idCursoRegime, designacao, regime
FROM curso_regime JOIN tb_curso USING(idCurso)
JOIN tb_regime USING (idRegime)
ORDER BY 2 -- Ascendente por defeito
LIMIT 0, 3
```

idCursoRegime	designacao	regime
1	CET de Multimédia	Laboral
6	CET de Multimédia	Pós laboral
4	CET de Redes	Laboral
2	CET de SI	Laboral
7	CET de SI	Pós laboral
3	EFA de Multimédia	Laboral
5	EFA de SI	Laboral

LIMIT

Os 3 primeiros a contar do segundo registo

idCursoRegime	designacao	regime
6	CET de Multimédia	Pós laboral
4	CET de Redes	Laboral
2	CET de SI	Laboral

```
SELECT idCursoRegime, designacao, regime
FROM curso_regime JOIN tb_curso USING(idCurso)
JOIN tb_regime USING (idRegime)
ORDER BY 2 -- Ascendente por defeito
LIMIT 1, 3
```

idCursoRegime	designacao	regime
1	CET de Multimédia	Laboral
6	CET de Multimédia	Pós laboral
4	CET de Redes	Laboral
2	CET de SI	Laboral
7	CET de SI	Pós laboral
3	EFA de Multimédia	Laboral
5	EFA de SI	Laboral



Exercício 12

-- Apresente os 3 últimos registos por ordem descendente

idCursoRegime	designacao	regime
5	EFA de SI	Laboral
3	EFA de Multimédia	Laboral
2	CET de SI	Laboral

```
USE formacao;  
  
SELECT idCursoRegime, designacao, regime  
FROM curso_regime JOIN tb_curso USING(idCurso)  
JOIN tb_regime USING (idRegime)  
ORDER BY 2 DESC  
LIMIT 0, 3
```



VIEWS

Como deve ter notado, temos de repetir o código SQL com as junções das tabelas sempre que pretendermos obter informação relacionada com elas. Mas podemos simplificar.

idRegime	idCurso	idCursoRegime	designacao	regime
1	1	1	CET de Multimédia	Laboral
1	2	2	CET de SI	Laboral
1	3	3	EFA de Multimédia	Laboral
1	4	4	CET de Redes	Laboral
1	5	5	EFA de SI	Laboral
2	1	6	CET de Multimédia	Pós laboral
2	2	7	CET de SI	Pós laboral

```
USE formacao;
```

```
DROP VIEW IF EXISTS v1_curso_regime;
```

```
CREATE VIEW v1_curso_regime AS
```

```
SELECT * FROM
```

```
curso_regime JOIN tb_curso USING(idCurso) JOIN  
tb_regime USING (idRegime);
```

```
SELECT * FROM v1_curso_regime;
```



VIEWS

Como deve ter notado, temos de repetir o código SQL com as junções das tabelas sempre que pretendermos obter informação relacionada com elas.

Mas podemos simplificar.

```
USE formacao;
```

```
SHOW tables;
```

Tables_in_formacao

curso_regime

tb_cidade

tb_curso

tb_emprego

tb_estadocivil

tb_formando

tb_genero

tb_habilitacao

tb_regime

v1_curso_regime



Mais SQL

Vamos aproveitar a VIEW que está guardada de modo a explorarmos mais funcionalidade de SQL.

idRegime	idCurso	idCursoRegime	designacao	regime
1	1	1	CET de Multimédia	Laboral
1	2	2	CET de SI	Laboral
1	3	3	EFA de Multimédia	Laboral
1	4	4	CET de Redes	Laboral
1	5	5	EFA de SI	Laboral
2	1	6	CET de Multimédia	Pós laboral
2	2	7	CET de SI	Pós laboral



Mais SQL

Selecionar da v1_curso_regime o idCursoRegime, a designacao e o regime

idCursoRegime	designacao	regime
1	CET de Multimédia	Laboral
2	CET de SI	Laboral
3	EFA de Multimédia	Laboral
4	CET de Redes	Laboral
5	EFA de SI	Laboral
6	CET de Multimédia	Pós laboral
7	CET de SI	Pós laboral

```
SELECT idCursoRegime, designacao, regime  
FROM v1_curso_regime;
```



Mais SQL - AS

Alterar o nome da coluna designacao para curso

idCursoRegime	curso	regime
1	CET de Multimédia	Laboral
2	CET de SI	Laboral
3	EFA de Multimédia	Laboral
4	CET de Redes	Laboral
5	EFA de SI	Laboral
6	CET de Multimédia	Pós laboral
7	CET de SI	Pós laboral

```
SELECT idCursoRegime, designacao AS curso, regime  
FROM v1_curso_regime;
```



Mais SQL - AS

Também podemos “alterar” o nome da VIEW ou tabela e usá-la no SELECT

idCR	curso	regime
1	CET de Multimédia	Laboral
2	CET de SI	Laboral
3	EFA de Multimédia	Laboral
4	CET de Redes	Laboral
5	EFA de SI	Laboral
6	CET de Multimédia	Pós laboral
7	CET de SI	Pós laboral

```
USE formacao;
```

```
SELECT v1.idCursoRegime AS idCR, designacao AS curso, regime  
FROM v1_curso_regime AS v1;
```



Mais SQL - DISTINCT

Selecionar apenas os regimes sem duplicados

regime
Laboral
Pós laboral

idCursoRegime	curso	regime
1	CET de Multimédia	Laboral
2	CET de SI	Laboral
3	EFA de Multimédia	Laboral
4	CET de Redes	Laboral
5	EFA de SI	Laboral
6	CET de Multimédia	Pós laboral
7	CET de SI	Pós laboral

```
SELECT DISTINCT regime  
FROM v1_curso_regime;
```



Mais SQL - DISTINCT

Selecionar apenas os regimes sem duplicados

regime
Laboral
Pós laboral

```
SELECT DISTINCT regime  
FROM v1_curso_regime;
```

Regimes dos cursos
Laboral
Pós laboral

```
SELECT DISTINCT regime AS 'Regimes dos cursos'  
FROM v1_curso_regime;
```



Mais SQL - WHERE

Selecionar apenas os registos com curso de 'CET de Multimédia'

idCursoRegime	designacao	regime
1	CET de Multimédia	Laboral
6	CET de Multimédia	Pós laboral

```
SELECT idCursoRegime, designacao, regime  
FROM v1_curso_regime  
WHERE designacao LIKE 'CET de Multimédia';
```



Mais SQL - WHERE

Selecionar apenas os registos com curso de 'CET'

idCursoRegime	designacao	regime
1	CET de Multimédia	Laboral
2	CET de SI	Laboral
4	CET de Redes	Laboral
6	CET de Multimédia	Pós laboral
7	CET de SI	Pós laboral

```
SELECT idCursoRegime, designacao, regime  
FROM v1_curso_regime  
WHERE designacao LIKE 'CET%';
```




Mais SQL - WHERE

Selecionar apenas os registos com curso de 'SI'

idCursoRegime	designacao	regime
2	CET de SI	Laboral
5	EFA de SI	Laboral
7	CET de SI	Pós laboral

```
-- SELECT  
USE formacao;  
  
SELECT idCursoRegime, designacao, regime  
FROM v1_curso_regime  
WHERE designacao LIKE '%SI';
```



Mais SQL - WHERE

Selecionar apenas os registos com curso de 'SI' em regime Laboral

idCursoRegime	designacao	regime
2	CET de SI	Laboral
5	EFA de SI	Laboral

```
SELECT idCursoRegime, designacao, regime  
FROM v1_curso_regime  
WHERE designacao LIKE '%SI' AND regime LIKE 'Laboral';
```



Mais SQL - WHERE

Selecionar apenas os registos com curso de 'EFA' ou em regime Pós Laboral

idCursoRegime	designacao	regime
3	EFA de Multimédia	Laboral
5	EFA de SI	Laboral
6	CET de Multimédia	Pós laboral
7	CET de SI	Pós laboral

```
SELECT idCursoRegime, designacao, regime  
FROM v1_curso_regime  
WHERE designacao LIKE '%EFA%' OR regime LIKE 'Pós Laboral';
```



Mais SQL - WHERE

Quantos cursos temos na tabela VIEW v1_curso_regime?

Total de cursos associados a regimes

7

idCursoRegime	designacao	regime
1	CET de Multimédia	Laboral
2	CET de SI	Laboral
3	EFA de Multimédia	Laboral
4	CET de Redes	Laboral
5	EFA de SI	Laboral
6	CET de Multimédia	Pós laboral
7	CET de SI	Pós laboral

```
SELECT COUNT(idCursoRegime) AS 'Total de cursos associados a regimes'  
FROM v1_curso_regime
```



Mais SQL - WHERE

E agora uma mais difícil???

Qual é o curso com idCursoRegime mais elevado?

idCursoRegime	designacao	regime
1	CET de Multimédia	Laboral
2	CET de SI	Laboral
3	EFA de Multimédia	Laboral
4	CET de Redes	Laboral
5	EFA de SI	Laboral
6	CET de Multimédia	Pós laboral
7	CET de SI	Pós laboral

idCursoRegime	designacao
7	CET de SI

```
SELECT idCursoRegime, designacao  
FROM v1_curso_regime WHERE idCursoRegime = 7;
```



Mais SQL - WHERE

E agora uma mais difícil???

Qual é o curso com idCursoRegime mais elevado?

idCursoRegime	designacao	regime
1	CET de Multimédia	Laboral
2	CET de SI	Laboral
3	EFA de Multimédia	Laboral
4	CET de Redes	Laboral
5	EFA de SI	Laboral
6	CET de Multimédia	Pós laboral
7	CET de SI	Pós laboral

idCursoRegime	designacao
7	CET de SI

```
SELECT idCursoRegime, designacao  
FROM v1_curso_regime WHERE idCursoRegime = 7;
```



Mais SQL - WHERE

E agora uma mais difícil???

Qual é o curso com idCursoRegime mais elevado?

idCursoRegime	designacao	regime
1	CET de Multimédia	Laboral
2	CET de SI	Laboral
3	EFA de Multimédia	Laboral
4	CET de Redes	Laboral
5	EFA de SI	Laboral
6	CET de Multimédia	Pós laboral
7	CET de SI	Pós laboral

MAX(idCursoRegime)

7

```
SELECT MAX(idCursoRegime) FROM v1_curso_regime;
```



Mais SQL - WHERE

E agora uma mais difícil???

Qual é o curso com idCursoRegime mais elevado?

idCursoRegime	designacao	regime
1	CET de Multimédia	Laboral
2	CET de SI	Laboral
3	EFA de Multimédia	Laboral
4	CET de Redes	Laboral
5	EFA de SI	Laboral
6	CET de Multimédia	Pós laboral
7	CET de SI	Pós laboral

MAX(idCursoRegime)

7

```
SELECT MAX(idCursoRegime) FROM v1_curso_regime;
```

```
SELECT idCursoRegime, designacao  
FROM v1_curso_regime WHERE idCursoRegime = ();
```




Mais SQL - WHERE

E agora uma mais difícil???

Qual é o curso com idCursoRegime mais elevado?

idCursoRegime	designacao
7	CET de SI

```
SELECT idCursoRegime, designacao  
FROM v1_curso_regime  
WHERE idCursoRegime = (SELECT MAX(idCursoRegime) FROM v1_curso_regime);
```



Exercício 13 – Qual é o curso com o idCursoRegime mais baixo?

idCursoRegime	designacao
1	CET de Multimédia

```
USE foreacao;  
SELECT idCursoRegime, @designacao  
FROM v1_curso_regime  
WHERE idCursoRegime = (SELECT MIN(idCursoRegime) FROM v1_curso_regime);
```



Exercício 14 – Registe o primeiro formando na tabela tb_formando

Field	Type	Null	Key	Default	Extra
idFormando	int(11)	NO	PRI	null	auto_increment
nome	text	NO		null	
apelido	text	NO		null	
dataNascimento	date	YES		null	
telemovel	text	YES		null	

idFormando	nome	apelido	dataNascimento	telemovel
1	Ana	Moreira	Wed Jun 20 1973 00:00:00 GMT+0100 (Western European Standard Time)	919730620

SQL

Vamos associar o formando às restantes tabelas?

idFormando	nome	apelido	dataNascimento	telemovel
1	Ana	Moreira	Wed Jun 20 1973 00:00:00 GMT+0100 (Western European Standard Time)	919730620

idCidade	cidade
1	Coimbra
2	Faro
3	Setúbal
4	Aveiro
5	Porto

idEmprego	emprego
1	Empregado
2	Desempregado
3	Estudante
4	Reformado

idEstadoCivil	estadoCivil
1	Solteiro
2	Casado
3	Comunhão
4	Divorciado
5	Viúvo

idHabilitacao	habilitacao
1	Básico
2	Secundário
3	Superior

idEstadoCivil	estadoCivil
1	Solteiro
2	Casado
3	Comunhão
4	Divorciado
5	Viúvo



SQL

Vamos associar o formando às restantes tabelas?

Para o fazermos podemos seguir 2 caminhos.

1. Criar uma tabela e associar tudo o que diz respeito ao formando
2. Criar uma tabela para cada associação.

Qual é a opção mais correta? A primeira ou a segunda?

Como acontece na maioria dos casos em SQL, a resposta tem de ser dada com outra pergunta.

SQL

Qual é a opção mais correta? A primeira ou a segunda?

Como acontece na maioria dos casos em SQL, a resposta tem de ser dada com outra pergunta.

Temos de perguntar a cada tabela se pretendemos guardar o histórico do formando ou também se para cada formando pode existir mais do que uma ocorrência.

SQL

Qual é a opção mais correta? A primeira ou a segunda?

Como acontece na maioria dos casos em SQL, a resposta tem de ser dada com outra pergunta.

Temos de perguntar a cada tabela se pretendemos guardar o histórico do formando ou também se para cada formando pode existir mais do que uma ocorrência.

SQL

Vamos associar o formando às restantes tabelas?

idFormando	nome	apelido	dataNascimento	telemovel
1	Ana	Moreira	Wed Jun 20 1973 00:00:00 GMT+0100 (Western European Standard Time)	919730620

idCidade	cidade
1	Coimbra
2	Faro
3	Setúbal
4	Aveiro
5	Porto

idEmprego	emprego
1	Empregado
2	Desempregado
3	Estudante
4	Reformado

idEstadoCivil	estadoCivil
1	Solteiro
2	Casado
3	Comunhão
4	Divorciado
5	Viúvo

idHabilitacao	habilitacao
1	Básico
2	Secundário
3	Superior

idEstadoCivil	estadoCivil
1	Solteiro
2	Casado
3	Comunhão
4	Divorciado
5	Viúvo



SQL

Tabela cidade:

- Pode um formando estar ou ter estado associado a mais do que uma cidade e essa informação é importante para mim?
- Caso a resposta seja negativa, podemos adicionar a coluna idFormando à tabela do formando ou à nova tabela de atributos do formando que pode ser criada.
- Esta resposta é relativa porque no momento em que estamos a criar a estrutura pode até nem ser. Mas depois com o tempo....

idFormando	nome	apelido
1	Ana	Moreira

idCidade	cidade
1	Coimbra
2	Faro
3	Setúbal
4	Aveiro
5	Porto

SQL

Tabela cidade:

- Vamos então admitir que pode haver necessidade de haver vários registos de cidade para o mesmo formando.
- Criamos então uma nova tabela para guardar esta associação.

Field	Type	Null	Key	Default	Extra
idFormandoCidade	int(11)	NO	PRI	null	auto_increment
idFormando	int(11)	YES		null	
idCidade	int(11)	YES		null	
dataRegisto	timestamp	NO		current_timestamp()	on update current_timestamp()

SQL

Field	Type	Null	Key	Default	Extra
idFormandoCidade	int(11)	NO	PRI	null	auto_increment
idFormando	int(11)	YES		null	
idCidade	int(11)	YES		null	
dataRegisto	timestamp	NO		current_timestamp()	on update current_timestamp()

```
DROP TABLE IF EXISTS tb_formando_cidade;
CREATE TABLE tb_formando_cidade(
  idFormandoCidade INT AUTO_INCREMENT PRIMARY KEY,
  idFormando INT,
  idCidade INT,
  dataRegisto TIMESTAMP
);
DESCRIBE tb_formando_cidade;
```



SQL

Tabela formando_cidade

Vamos lá associar a Ana Moreira à cidade do Porto.

idCidade	cidade
5	Porto

```
USE formacao;  
SELECT * FROM tb_cidade WHERE cidade LIKE 'Porto'
```

idFormando	nome	apelido
1	Ana	Moreira

```
USE formacao;  
SELECT idFormando, nome, apelido FROM tb_formando  
WHERE nome LIKE 'Ana' AND apelido LIKE 'Moreira';
```

SQL

Vamos lá associar a Ana Moreira à cidade do Porto.

idFormandoCidade	idFormando	idCidade	dataRegisto
1	1	5	Sun May 21 2023 18:33:40 GMT+0100 (Western European Summer Time)

idFormando	nome	apelido
1	Ana	Moreira

idCidade	cidade
5	Porto

```
USE formacao;  
INSERT INTO tb_formando_cidade (idFormando, idCidade)  
VALUES (1, 5);  
  
SELECT * FROM tb_formando_cidade;
```



SQL

nome	apelido	cidade	telemovel
Ana	Moreira	Porto	919730620

idFormandoCidade	idFormando	idCidade
1	1	5

```
USE formacao;  
SELECT nome, apelido, cidade, telemovel  
FROM tb_formando_cidade  
JOIN tb_formando USING(idFormando)  
JOIN tb_cidade USING(idCidade);
```



Exercício 15

```
DESCRIBE tb_formando_genero;
```

Field	Type	Null	Key	Default	Extra
idFormandoGenero	int(11)	NO	PRI	null	auto_increment
idFormando	int(11)	YES		null	
idGenero	int(11)	YES		null	
dataRegisto	timestamp	NO		current_timestamp()	on update current_timestamp()



Exercício 16.1

```
DESCRIBE tb_formando_genero;
```

Field	Type	Null	Key	Default	Extra
idFormandoGenero	int(11)	NO	PRI	null	auto_increment
idFormando	int(11)	YES		null	
idGenero	int(11)	YES		null	
dataRegisto	timestamp	NO		current_timestamp()	on update current_timestamp()



Exercício 16.2

```
SELECT * FROM tb_formando_cidade;
```

idFormandoCidade	idFormando	idCidade	dataRegisto
1	1	5	Sun May 21 2023 18:33:40 GMT+0100 (Western European Summer Time)



Exercício 16.3

nome	apelido	genero	telemovel
Ana	Moreira	Feminino	919730620



Exercício 16.4

nome	apelido	genero	cidade	telemovel
Ana	Moreira	Feminino	Porto	919730620



Exercício 17.1

```
DESCRIBE tb_formando_estadoCivil;
```

Field	Type	Null	Key	Default	Extra
idFormandoEstadoCivil	int(11)	NO	PRI	null	auto_increment
idFormando	int(11)	YES		null	
idEstadoCivil	int(11)	YES		null	
dataRegisto	timestamp	NO		current_timestamp()	on update current_timestamp()



Exercício 17.2

idFormandoEstadoCivil	idFormando	idEstadoCivil	dataRegisto
1	1	1	Sun May 21 2023 19:01:56 GMT+0100 (Western European Summer Time)
2	1	2	Sun May 21 2023 19:01:56 GMT+0100 (Western European Summer Time)



Exercício 17.3

nome	apelido	estadoCivil
Ana	Moreira	Solteiro
Ana	Moreira	Casado



Exercício 17.4

nome	apelido	estadoCivil
Ana	Moreira	Solteiro,Casado

-- Dica

```
SELECT nome, apelido, genero, cidade, estadoCivil, telemovel
```

nome	apelido	genero	cidade	estadoCivil	telemovel
Ana	Moreira	Feminino	Porto	Solteiro,Casado	919730620



Exercício 17.5

nome	apelido	genero	cidade	estadoCivil	telemovel
Ana	Moreira	Feminino	Porto	Solteiro,Casado	919730620



Exercício 18.1

```
DESCRIBE tb_formando_habilitacao;
```

Field	Type	Null	Key	Default	Extra
idFormandoHabilitacao	int(11)	NO	PRI	null	auto_increment
idFormando	int(11)	YES		null	
idHabilitacao	int(11)	YES		null	
dataRegisto	timestamp	NO		current_timestamp()	on update current_timestamp()

Exercício 18.2

idFormandoHabilitacao	idFormando	idHabilitacao	dataRegisto
1	1	1	Sun May 21 2023 19:31:38 GMT+0100 (Western European Summer Time)



Exercício 18.3

nome	apelido	habilitacao
Ana	Moreira	Básico



Exercício 19.1

```
DESCRIBE tb_formando_emprego;
```

Field	Type	Null	Key	Default	Extra
idFormandoEmprego	int(11)	NO	PRI	null	auto_increment
idFormando	int(11)	YES		null	
idEmprego	int(11)	YES		null	
dataRegisto	timestamp	NO		current_timestamp()	on update current_timestamp()



Exercício 19.2

```
SELECT * FROM tb_formando_emprego;
```

idFormandoEmprego	idFormando	idEmprego	dataRegisto
1	1	1	Sun May 21 2023 19:37:54 GMT+0100 (Western European Summer Time)



Subrotinas

- Procedimentos



Subrotinas



Exercícios