

# Sistemas Transaccionales y Bases de Datos

25 de Septiembre, 2023



red.es

CRN  
Digital



Barrabés

The Valley

"El FSE invierte en tu futuro"

Fondo Social Europeo

1. Introducción a las Bases de Datos
  1. Los 3 Mundos
  2. Entidades y Atributos
  3. El Mundo de las Representaciones
2. Conceptos Básicos
  1. Diseño de Bases de Datos
  2. Manejar Bases de Datos
  3. Modelos, administración y lenguajes de BBDD
  3. Lenguajes de acceso a BBDD 
  4. Modelos de bases de datos
  5. Resumen

# Los 3 Mundos



# Los Tres Mundos

## Definiciones

Distinguiremos tres ámbitos diferentes:

El mundo real con los objetos de nuestro interés.

El mundo de las conceptualizaciones lógicas

El mundo de las representaciones informáticas



# Los Tres Mundos

## El Mundo Real

El mundo real, la parte de la realidad que nos interesa, es lo que percibimos con nuestros sentidos y está compuesto por objetos concretos, físicos o no.

## El Mundo Conceptual

El conjunto de los conocimientos obtenidos a partir de la observación de un mundo real se denomina mundo conceptual o mundo de las concepciones.

El proceso de observación/abstracción es básicamente un proceso para modelizar la estructura, las propiedades y el funcionamiento de la realidad.

La información es un conocimiento transmisible, es decir, que se puede representar.

# Los Tres Mundos

## El Mundo de las Representaciones

Para poder trabajar con **conocimientos** y poderlos comunicar, necesitamos proyectar los pensamientos en el exterior representante físicamente de alguna manera. Este es el **Mundo de las Representaciones**.

Damos el nombre de **datos** a las representaciones físicas de los conocimientos que tenemos de los objetos del mundo real.

**El paso de los conocimientos a los datos, o de una concepción a una representación informática, es un proceso humano.**

# Los Tres Mundos

What now?

Acabamos de ver el camino que nos lleva de la realidad a los conocimientos, y de éstos a los datos o representaciones. Pero nos hará falta poder interpretar la representación. El proceso inverso al de representación se conoce como interpretación.

# Entidades y Atributos



# Primeras Definiciones - Vuelta al comienzo

## Entidades, atributos y valores

En términos **lingüísticos**: una información (un conocimiento elemental) se puede expresar con un sujeto (el modelo concreto) y un predicado ("salió al mercado en 1974"). El predicado es formado por el verbo y el complemento.

Desde un punto de vista informático:

- ❖ Entidades: los **objetos** que conceptualizamos como distinguibles unos de otros (es decir, **que son identificables**) y de los que nos interesan algunas propiedades.

El predicado es la propiedad descrita, y a sus dos partes:

- ❖ verbo: **atributo** (año de producción)
- ❖ complemento: **valor** (1974), respectivamente.

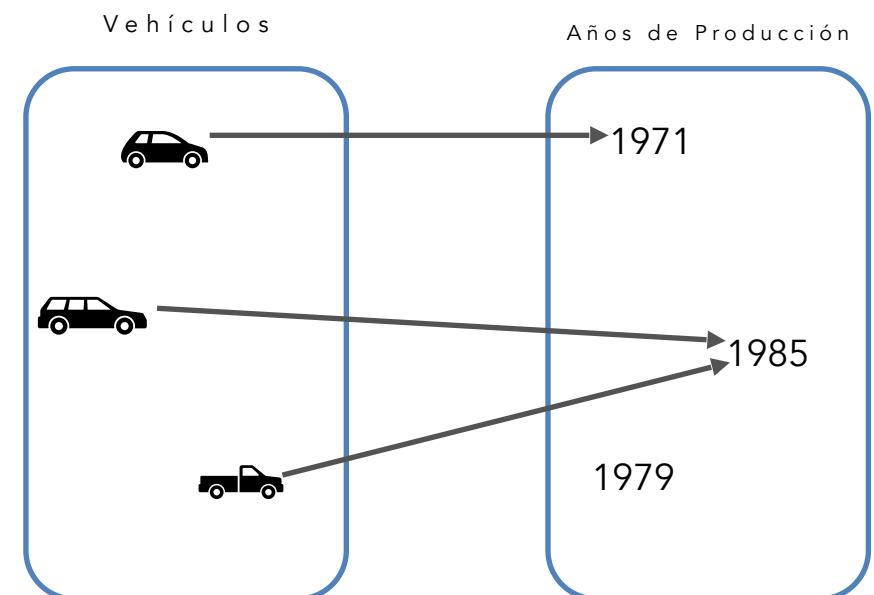
Componentes de información elemental		
Sujeto	Predicado	
Entidad	Atributo	Valor
El VW Golf	salió al mercado en	1974

# El Marco de la Teoría de Conjuntos

## Entidades

Usando la teoría de conjuntos, podemos ver la correspondencia entre los estudiantes y los años como una **aplicación** (matemática) del conjunto de los vehículos sobre el conjunto de los años:

- Cada modelo de coche tiene un solo año de producción.
- Diferentes modelos pueden tener el mismo año de producción.



# El Marco de la Teoría de Conjuntos

## Definiciones

El conjunto de todos los valores válidos, o legales, que puede llegar a tener un atributo, recibe el nombre de **dominio del atributo**.

Puede ocurrir que el valor de un atributo determinado de alguna entidad individual sea desconocido o no exista. Entonces diremos que el dominio de ese atributo acepta el valor nulo.

Los atributos que concebimos como aplicaciones inyectivas<sup>1</sup> se denominan **identificadores**.

Todo atributo o conjunto de atributos que permite identificar las entidades individuales recibe el nombre de **clave**.

Dado que el atributo es una aplicación entre conjuntos, a cada entidad le puede corresponder como máximo un solo valor. En consecuencia, un **atributo no podrá ser multivalor** (o multivaluado).

(1) Una aplicación inyectiva es aquella en la que cada elemento del conjunto imagen le corresponde un elemento del conjunto origen como máximo.

# El Mundo de las Representaciones

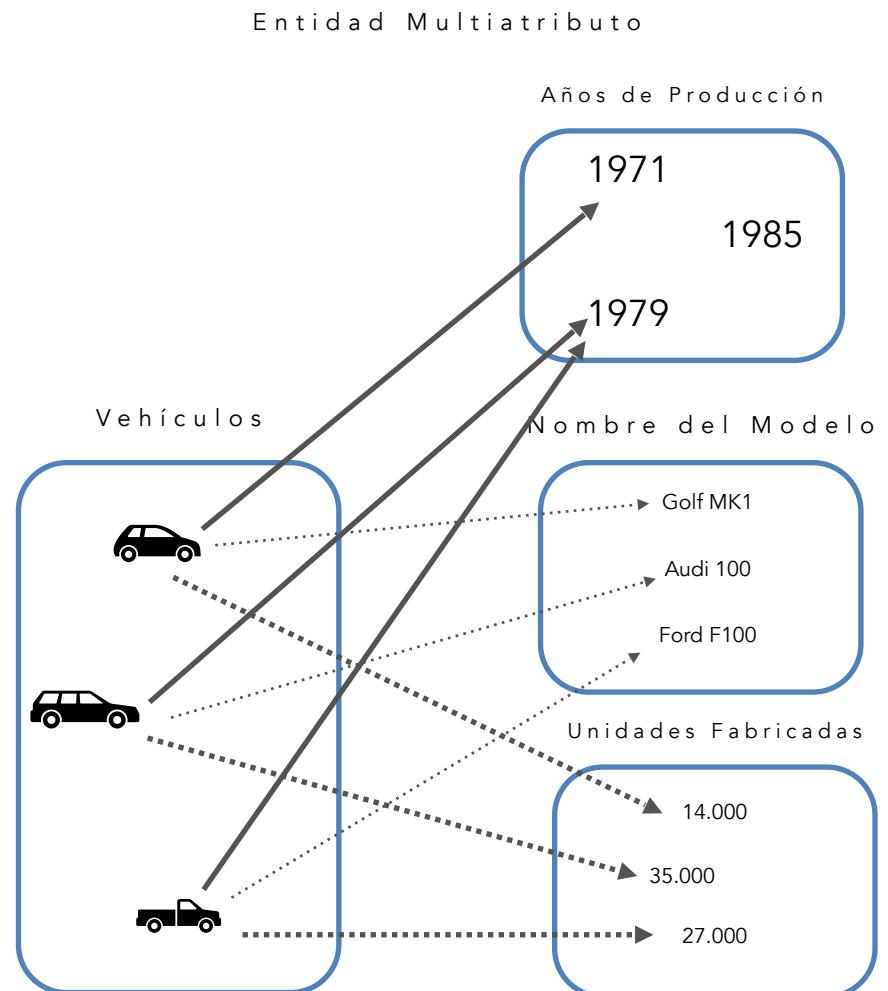


# El Mundo de las Representaciones

## Representación Gráfica

La figura de la derecha es una representación gráfica, no informatizada, de la información de los vehículos, utilizando la representación clásica de las aplicaciones en la teoría de conjuntos, para una entidad multiatributo.

**Con tantas flechas y conjuntos, no resulta muy cómodo para ser procesada...**



# El Mundo de las Representaciones

## Representación Tabular

Resulta mucho más sencilla una representación tabular con una fila para cada entidad individual y una columna para cada atributo.

Vehículo	Marca del Vehículo	Nombre del Vehículo	Año de Producción	Unidades Fabricadas
	Volkswagen	Golf Mk1	1971	14.000
	Toyota	Corolla Wagon	1979	35.000
	Ford	F100	1979	27.000

# El Mundo de las Representaciones

## Archivos de Datos

Un archivo de datos es una representación informática equivalente a la representación tabular

- La representación de una entidad, el equivalente a una fila de la tabla, recibe el nombre de registro.
- La representación del valor de un atributo de una entidad se denomina campo.

El conjunto de campos constituye el registro, y el conjunto de registros constituye el archivo.

Marca del Vehículo	Nombre del Vehículo	Año de Producción	Unidades Fabricadas
Volkswagen	Golf Mk1	1971	14.000
Toyota	Corolla Wagon	1979	35.000
Ford	F100	1979	27.000

Nombre/Cabecera de los Campos

Registros

# El Mundo de las Representaciones

## Ficheros interrelacionados

Hasta nueva orden, vamos a definir una base de datos (BBDD) como un conjunto de ficheros de datos interrelacionados.

Supongamos que los tipos de objetos de nuestro interés son estudiantes, asignaturas y profesores.

Podremos representar estas entidades mediante tres ficheros, uno para cada entidad, con los campos correspondientes a los atributos. Pero falta la información que permite interrelacionar las entidades entre sí.

- ❖ Todo estudiante puede cursar más de una asignatura y toda asignatura puede ser cursada por muchos estudiantes.
- ❖ Toda asignatura es dada por un solo profesor, pero cada profesor puede dar varias asignaturas.

Proponed un ejemplo de estos 3 ficheros y sus posibles relaciones - 10 min

# El Mundo de las Representaciones

## Acceso a los datos

Hay dos formas básicas de acceso a los datos

- ❖ **Acceso secuencial:** el acceso a un registro presupone el acceso previo a todos los registros anteriores.
- ❖ **Acceso directo:** el acceso a un registro no requiere el acceso previo a todos los registros anteriores.

Adicionalmente, distinguimos entre:

- ❖ **Acceso por valor:** el acceso por valor nos lleva al registro en función del valor de alguno de sus atributos, sin tener en cuenta la posición que ocupa el registro.
- ❖ **Acceso por posición:** el acceso por posición nos lleva a un lugar - una posición - donde hay un registro de datos, sin tener en cuenta el contenido.

# El Mundo de las Representaciones

## Acceso a los datos

Distinguiremos dos niveles en el mundo de las representaciones informáticas

- ❖ **Nivel lógico:** en este nivel, nos oculta los detalles de cómo se almacenan los datos, cómo se mantienen y cómo se accede físicamente a ellos.

En este nivel sólo se habla de entidades, atributos y reglas de integridad, no necesitamos conocer ni ver, la realización física que puede constar, por ejemplo, de encadenamientos de registros, marcas separadoras entre campos, compresión de datos o índices (PK, FK).

- ❖ **Nivel físico:** entramos en el nivel físico cuando tenemos que considerar las posibles realizaciones físicas.

Quizá nos interese poder describir elementos de nivel físico como los índices que tendremos, cómo y dónde queremos que se agrupen físicamente los registros o de qué tamaño deben ser las páginas.

# Diseño de Bases de Datos



# Diseño de Bases de Datos

## Arquitectura ANSI/SPARC

Una nueva forma de pensar las BBDD

De acuerdo con la arquitectura **ANSI/SPARC**, debía haber tres niveles de esquemas (tres niveles de abstracción).

La idea básica de **ANSI/SPARC** consistía en descomponer el **nivel lógico** en dos:

- ❖ el **nivel externo**
- ❖ el **nivel conceptual**

Denominaremos nivel interno lo que hasta ahora hemos denominado nivel físico.

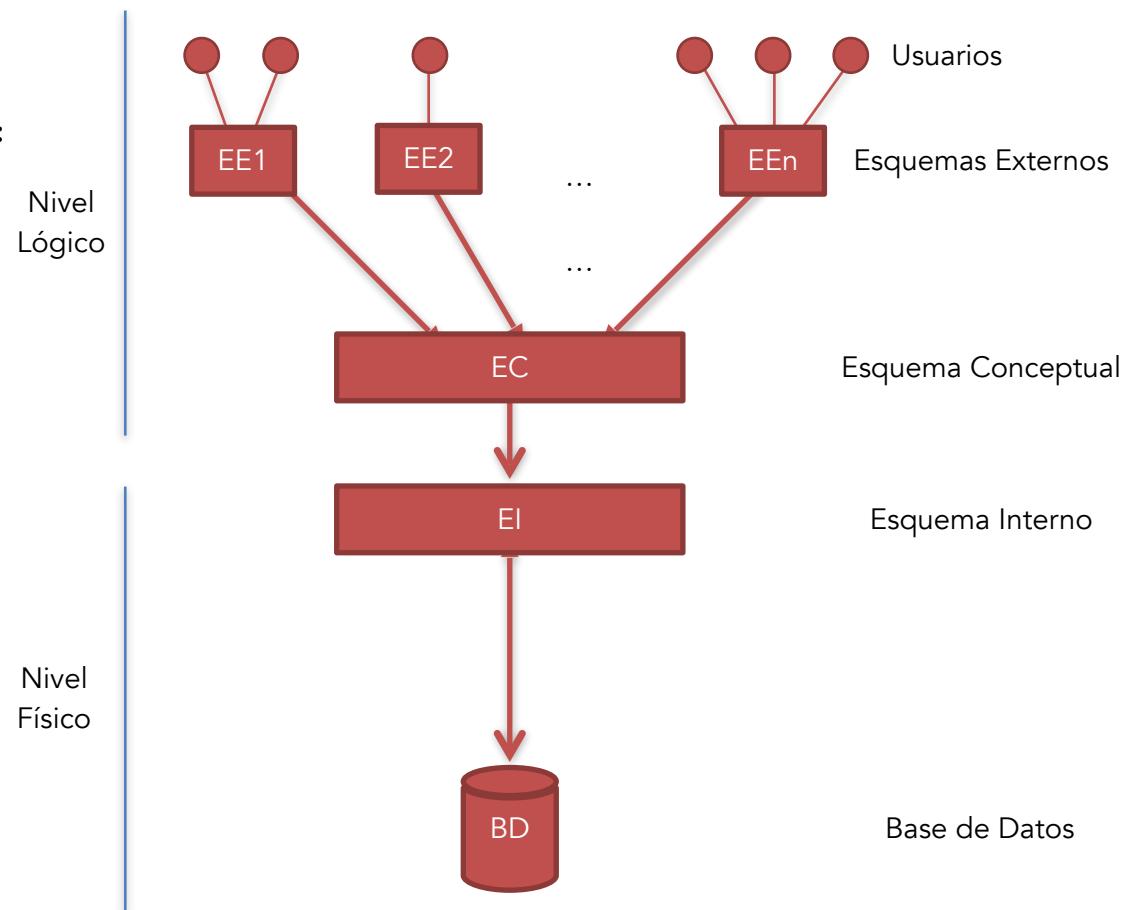
Anteriormente	ANSI / SPARC
Lógico	Externo Conceptual
Físico	Interno

# ANSI/SPARC

## Los tres niveles

Estructuralmente, encontraremos los siguientes elementos en los niveles:

- ❖ En el **nivel externo** se sitúan las diferentes visiones lógicas que los procesos usuarios (programas de aplicación y usuarios directos) tendrán de las partes de la BD que utilizarán. A estas visiones las llamaremos esquemas externos.
- ❖ En el **nivel conceptual** hay una sola descripción lógica básica, única y global, que denominamos esquema conceptual, y que sirve de referencia para el resto de los esquemas.
- ❖ En el **nivel físico** hay una sola descripción física, que denominamos esquema interno.



# ANSI/SPARC

## Tipos de Esquemas

- ❖ **Esquema Conceptual:** en este esquema, escribiremos las entidades tipo, sus atributos, las interrelaciones y las restricciones o reglas de integridad.
- ❖ Al definir un **Esquema Externo**, se citarán sólo aquellos atributos y aquellas entidades que interesen; pudiendo definir datos derivados o redefinir una entidad para que las aplicaciones que utilizan este esquema externo crean que son dos, definir combinaciones de entidades, renombrar registros, etc.
- ❖ El **Esquema Interno** contiene la descripción de la organización física de la BD: caminos de acceso (índices, hashing, apuntadores, etc.), codificación de los datos, gestión del espacio, tamaño de la página, etc.

# ANSI/SPARC

## Independencia de los datos

Hay **independencia física** cuando los cambios en la organización física de la BD no afectan al mundo exterior (es decir, los programas usuarios o los usuarios directos).

Hay **independencia lógica** cuando los usuarios no se ven afectados por los cambios en el nivel lógico.

# Manejar Bases de Datos



# Bases de Datos

## Definiciones

Llamamos **Bases de Datos** a los conjuntos de ficheros interrelacionados, con estructuras complejas y que son compartidos por varios procesos de forma simultánea.

Una base de datos de un SI es la representación integrada de los conjuntos de entidades instancia correspondientes a las diferentes entidades tipo del SI y de sus interrelaciones.

Esta representación informática debe poder ser utilizada de forma compartida por muchos usuarios de distintos tipos.

El software especializado en la gestión de las bases de datos se denomina **Sistema de Gestión de Bases de Datos (SGBD)**

# Sistemas de Gestión de Bases de Datos



# Sistemas de Gestión de Bases de Datos

## Objetivos

- Permitir la realización de consultas no predefinidas y complejas.
- Ofrecer flexibilidad a los cambios (independencia).
- Facilitar la eliminación y la gestión de la redundancia.
- Asegurar el mantenimiento de la calidad de los datos (integridad).
- Gestionar la concurrencia de acceso a los datos.
- Proveer servicios de seguridad.

# Sistemas de Gestión de Bases de Datos

## Esquema de la BBDD

Los SGBD necesitan que les demos una descripción o **definición** de la BD.

Esta descripción recibe el nombre de **esquema** de la BD, y los SGBD la tendrán continuamente a su alcance.

El **nivel lógico** nos oculta los detalles de cómo se almacenan los datos, cómo se mantienen y cómo se accede físicamente a ellos. En este nivel sólo se habla de entidades, atributos y reglas de integridad.

Por cuestiones de rendimiento, nos podrá interesar describir elementos de **nivel físico** como qué índices tendremos, cómo y dónde queremos que se agrupen físicamente los registros, de qué tamaño deben ser las páginas, etc.

# Sistemas de Gestión de Bases de Datos

## Modelos de BBDD

El conjunto de componentes o herramientas conceptuales que un SGBD proporciona para modelar recibe el nombre de **modelo de BD**.

Algunos ejemplos de modelos de BBDD son el **modelo relacional**, el **modelo jerárquico**, el **modelo en red** y el **modelo relacional con objetos**.

Los modelos de BBDD nos proporcionan tres tipos de herramientas:

- ◆ Estructuras de datos con las que se puede construir la BBDD: tablas, árboles, etc.
- ◆ Diferentes tipos de restricciones (o reglas) de integridad que el SGBD tendrá que hacer cumplir a los datos: dominios, claves, etc.
- ◆ Una serie de operaciones para trabajar con los datos.

# Sistemas de Gestión de Bases de Datos

## Administración de bases de datos

Una empresa o institución que tenga SI construidos en torno a BD necesita que alguien lleve a cabo una serie de funciones centralizadas de gestión y administración, para asegurar que la explotación de la BD es la correcta.

Un administrador de base de datos se encarga de tareas como:

- ❖ Mantenimiento, administración y control de los esquemas.
- ❖ Asegurar la máxima disponibilidad de los datos (backups, logs, etc.)
- ❖ Resolución de emergencias.
- ❖ Vigilancia de la integridad y de la calidad de los datos.
- ❖ Diseño físico, estrategia de caminos de acceso y reestructuraciones.
- ❖ Control del rendimiento.
- ❖ Normativa y asesoramiento a los programadores y a los usuarios finales.
- ❖ Control y administración de la seguridad: autorizaciones, restricciones, etc.

# Modelos de Bases de Datos



# Lenguajes de acceso a BBDD



# Lenguajes

## Lenguajes

Para comunicarse con el SGBD, el usuario, ya sea un programa de aplicación o un usuario directo, se vale de un lenguaje.

Vamos a poder clasificarlos según dos criterios principales,

1. El tipo de usuarios para los que están pensados.
  2. El tipo de cosas que los usuarios deben poder expresar con ellos.
- 
1. Hay lenguajes especializados en la escritura de esquemas; es decir, en la descripción de la BD. Se conocen genéricamente como **DDL** o **Data Definition Language**.
  2. Otros lenguajes están especializados en la utilización de la BD (consultas y mantenimiento). Se conocen como **DML** o **Data Management Language**.

**Lo más frecuente es que el mismo lenguaje disponga de construcciones para las dos funciones, DDL y DML.**

Por ejemplo, SQL tiene instrucciones DDL (como CREATE TABLE), instrucciones DML como (SELECT) y adicionalmente instrucciones de control de entorno (p.e. COMMIT).

# Resumen



# Resumen

## Ventajas e Inconvenientes

Ventajas	Desventajas
Acceso rápido a los datos	Tamaño Creciente
Compartir datos remotamente	Posibles costes elevados de mantenimiento
Centralización de la información	Actualizaciones
Reducción de espacio físico necesario	Posibles fallos críticos
Mantenimiento y actualización sencillas	Necesidad de conexión permanente
Control de versiones	Problemas de permisos
Posibilidad de portabilidad	Necesidad de un control experto
Evitaremos datos duplicados o repetidos	
Son dinámicas	



GOBIERNO  
DE ESPAÑA

VICEPRESIDENCIA  
PRIMERA DEL GOBIERNO  
MINISTERIO  
DE ASUNTOS ECONÓMICOS  
Y TRANSFORMACIÓN DIGITAL

SECRETARÍA DE ESTADO  
DE DIGITALIZACIÓN  
E INTELIGENCIA ARTIFICIAL

red.es

Centro de  
Referencia Nacional  
en Comercio Electrónico  
y Marketing  
**CRN**  
Digital



UNIÓN EUROPEA

*"El FSE invierte en tu futuro"*

Fondo Social Europeo

  
Barrabés

 The Valley

# Sistemas Transaccionales y Bases de Datos

26 de Septiembre, 2023



red.es

"El FSE invierte en tu futuro"

Fondo Social Europeo

CRN  
Digital



Barrabés

The Valley

1. SQL - Introducción
2. Principales Queries



# SQL - Introducción



# SQL

## Objetivos

- ❖ Conocer los conceptos básicos del lenguaje estándar SQL.
- ❖ Definir una BD relacional.
- ❖ Saber introducir, borrar y modificar datos.
- ❖ Ser capaz de plantear cualquier tipo de consulta a la BD.

# SQL

## Cambios con respecto a ayer

C o l u m n a s

Marca del Vehículo	Nombre del Vehículo	Año de Producción	Unidades Fabricadas
Volkswagen	Golf Mk1	1971	14.000
Toyota	Corolla Wagon	1979	35.000
Ford	F100	1979	27.000

F i l a s

# SQL

## Estructura

Sentencia

<b>SELECT</b> codigo_producto, num_producto, tipo	Cláusula
<b>FROM</b> productos	Cláusula
<b>WHERE</b> precio > 1000.0	Cláusula

# Principales Queries



# Principales Queries

## Primeras funciones

- ❖ **CREATE** Para crear BD, tablas, dominios y aserciones.
- ❖ **ALTER** Para modificar tablas y dominios se utilice.
- ❖ **DROP** Para borrar BD, tablas, dominios y aserciones

# Principales Queries

## Primeros pasos - Creación de un Esquema

SQL estándar no dispone de ninguna sentencia de creación de BD.

En cambio, disponemos de una sentencia más potente que la de creación de BD: la sentencia de creación de esquemas llamada **CREATE SCHEMA**

```
CREATE SCHEMA {schema_name | AUTHORIZATION <usuario>}  
[<lista_elementos_esquema>];
```

La sentencia de creación de esquemas permite que un **conjunto de tablas** (*lista\_elementos\_esquema*) se pueda agrupar bajo un mismo **nombre** (*nombre\_esquema*) y que tengan un **propietario** (*usuario*).

# Principales Queries

## Primeros pasos - Eliminación de un Esquema

SQL estándar no dispone de ninguna sentencia de eliminación de BD.

En cambio, disponemos de una sentencia más potente que la de creación de BD: la sentencia de creación de esquemas llamada **DROP SCHEMA**

**DROP SCHEMA <schema\_name> {RESTRICT | CASCADE}**

La opción **RESTRICT** hace que el esquema sólo se pueda borrar si no contiene ningún elemento.

La opción **CASCADE** borra el esquema aunque no esté completamente vacío.

# Principales Queries

## Primeros pasos - Crear una tabla

```
CREATE TABLE <nombre_tabla>
(
    <definicion_columna>,
    [, <definicion_columna>, ...]
    [, <restricciones_tabla>]
);
```

Donde *definicion\_columna* es:

```
<nombre_columna> {<tipo_datos>|<dominio>} [<def_defecto>] [<restricciones_columna>]
```

# Principales Queries

## Tipos de Datos

¿Qué tipos de datos conocéis?

Tipo de datos	Descripción	Ejemplo
CHARACTER (longitud)	Cadenas de caracteres de longitud fija	CHAR(9) → 67531234P
CHARACTER VARYING (longitud)	Cadenas de caracteres de longitud variable	VARCHAR(20) → Juan Alberto
BIT (longitud)	Cadenas de bits de longitud fija	BIT(8) → B'10101010'
BIT VARYING (longitud)	Cadenas de bits de longitud variable	VARBIT(16) → B'10101010'
NUMERIC (precisión, escala)	Números decimales con tantos dígitos como indique la precisión y tantos decimales como indique la escala	NUMERIC(8,2) → 123456.78
DECIMAL (precisión, escala)	Números decimales con tantos dígitos como indique la precisión y tantos decimales como indique la escala	DECIMAL(8,2) → 123456.78
INTEGER	Números Enteros	INT → 12345321
SMALLINT	Números Enteros Pequeños(menores	SMALLINT → 2 (menor que $2^{15} = 32768$ )
REAL	Números con coma flotante con precisión predefinida	REAL → 3.463
DOUBLE PRECISION	Números con coma flotante con más precisión predefinida que la del tipo REAL	DOUBLE → 3.4632312
FLOAT (precisión)	Números con coma flotante con precisión especificada	FLOAT(2) → 3.23
DATE	Fechas. Se componen de: YEAR año, MONTH mes, DAY día	DATE → '2023-06-06'
TIME	Horas. Se componen de: : HOUR hora, MINUT minutos, SECOND segundos	TIME → '14:30:00'
TIMESTAMP	Fechas y horas. Se componen de: YEAR año, MONTH mes, DAY día, HOUR hora, MINUT minutos, SECOND segundos	TIMESTAMP → '2023-06-06 14:30:00'

# Principales Queries

## Valores por Defecto

Y los valores nulos?

**DEFAULT (<literal> |<funcion>|NULL)**

Para un empleado del cual todavía no se ha decidido cuánto ganará, se puede escoger que, de momento, tenga un sueldo de 0 euros (**DEFAULT 0.0**), o bien que tenga un sueldo con un valor nulo (**DEFAULT NULL**).

# Principales Queries

## Primeros pasos - Restricciones de Columna

¿Todos los valores tienen sentido en todas las columnas?

Restricción	Descripción
NOT NULL	La columna no puede tener valores nulos
UNIQUE	La columna no puede tener valores repetidos. Es una clave alternativa
PRIMARY KEY	La columna no puede tener valores repetidos ni nulos. Es la clave primaria
REFERENCES <i>&lt;nombre_tabla&gt;[(&lt;nombre_columna&gt;)]</i>	La columna es la clave foránea de la columna de la tabla especificada
CONSTRAINT [<nombre_restriccion>] CHECK (<condiciones>)	La columna tiene que cumplir las condiciones especificadas

# Principales Queries

## Primeros pasos - Restricciones de Tabla

¿Podemos controlar toda la tabla con alguna restricción?

Restricción	Descripción
UNIQUE <nombre_columna> [, <nombre_columna>...]	El conjunto de las columnas especificadas no puede tener valores repetidos.
PRIMARY KEY <nombre_columna> [, <nombre_columna>...]	El conjunto de las columnas especificadas no puede tener valores nulos ni repetidos.
FOREIGN KEY <nombre_columna> [, <nombre_columna>...] REFERENCES <nombre_tabla> [(<nombre_columna2> , <nombre_columna2>...)]	El conjunto de las columnas especificadas es una clave foránea que referencia la clave primaria formada por el conjunto de las columnas2 de la tabla dada. Si las columnas y las columnas2 se llaman exactamente igual, entonces no habría que poner columnas2.
CONSTRAINT [<nombre_restriccion>] CHECK (<condiciones>)	La tabla tiene que cumplir las condiciones especificadas

# Principales Queries

## Primeros pasos - Alteraciones de Tablas

Además de poder crear tablas, vamos a ser capaces de cambiar su estructura.

```
ALTER TABLE <nombre_tabla> | {<accion_modificar_columna>
                                <accion_modificar_restriccion_tabla>}
```

Donde tenemos las dos opciones:

**accion\_modificar\_columna:**

```
{ADD [COLUMN] <nombre_columna> <def_columna> |
 ALTER [COLUMN] <nombre_columna> {SET <def_defecto>| DROP DEFAULT}
 DROP [COLUMN] <nombre_columna> {RESTRICT | CASCADE} }
```

**accion\_modificar\_restriccion\_tabla:**

```
{ADD <def_restriccion> |
 DROP CONSTRAINT <nombre_restriccion> {RESTRICT | CASCADE} }
```

# Principales Queries

## Primeros pasos - Borrado de tablas

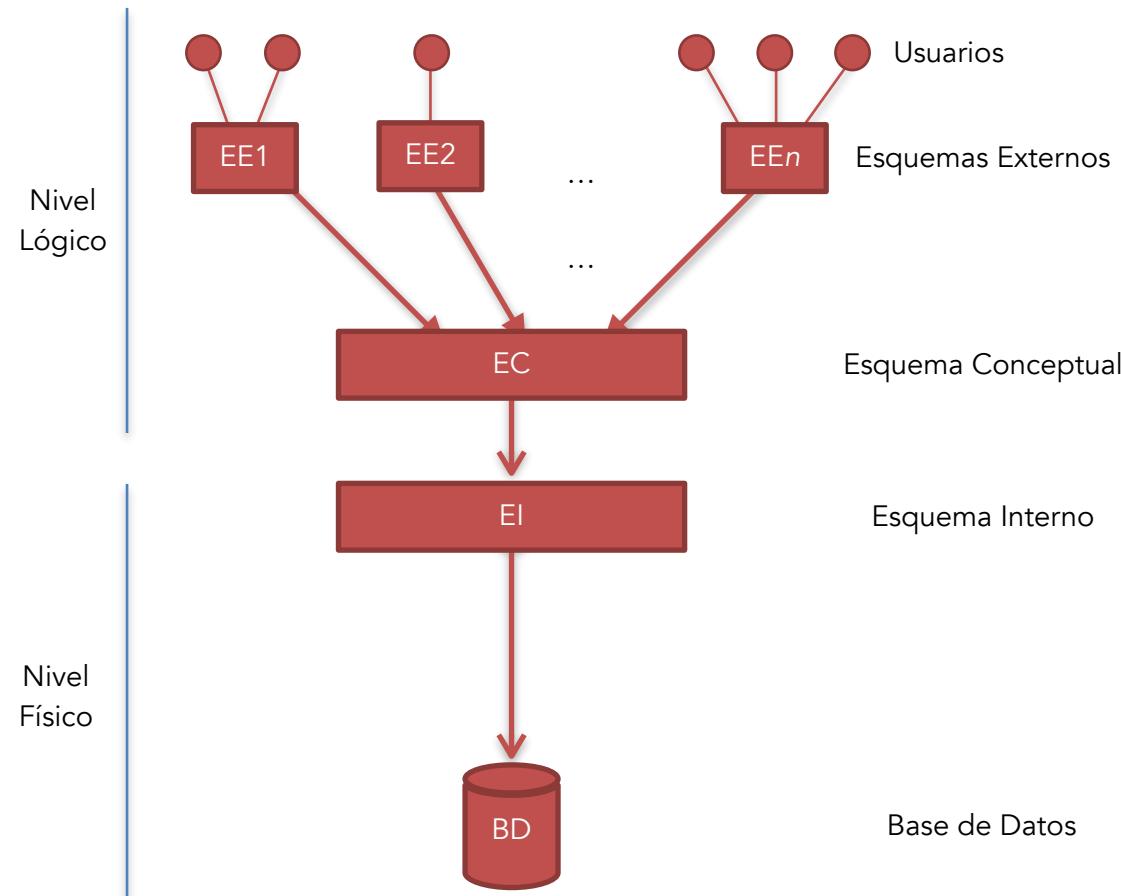
“Reestructuración”

```
DROP TABLE <nombre_tabla> {RESTRICT | CASCADE}
```

# Principales Queries

## Primeros pasos - Creación de visiones

ANSI/SPARC tenía 3 niveles, ¿dónde hemos estado trabajando?



**CREATE VIEW <nombre\_vista> [(lista\_consultas)] as (consulta)  
{WITH CHECK OPTION}**

**CREATE VIEW** estudiantes por asignatura (id\_asignatura,  
num\_estudiante)

(SELECT a.id\_asignatura, count(a.id\_estudiante) as  
num\_estudiante

FROM matriculados as a

GROUP BY a.id\_asignatura);

**DROP VIEW <nombre\_tabla> {RESTRICT | CASCADE}**

# Ejercicio 1



# Ejercicio 1

## Creación de nuestra primera Base de Datos

30 min

### Vamos a crear nuestra segunda Base de Datos

Vamos a crear una Base de Datos para una empresa.

Ha de tener como mínimo 4 tablas:

**Tabla Departamentos:** nombre del departamento, ciudad del departamento y teléfono del departamento

**Tabla Clientes:** código del cliente, nombre del cliente, nif, dirección, ciudad y teléfono del cliente

**Tabla Empleados:** código del empleado, nombre, apellido, sueldo, nombre del departamento al que pertenece, ciudad del departamento y número del proyecto asignado.

**Tabla Proyectos:** código del proyecto, nombre del proyecto, precio, fecha de inicio, fecha prevista de finalización y fecha de finalización, código del cliente.

**Tabla Proyectos\_Empieados:** código del proyecto, código del empleado



GOBIERNO  
DE ESPAÑA

VICEPRESIDENCIA  
PRIMERA DEL GOBIERNO  
MINISTERIO  
DE ASUNTOS ECONÓMICOS  
Y TRANSFORMACIÓN DIGITAL

SECRETARÍA DE ESTADO  
DE DIGITALIZACIÓN  
E INTELIGENCIA ARTIFICIAL

red.es

Centro de  
Referencia Nacional  
en Comercio Electrónico  
y Marketing  
**CRN**  
Digital



UNIÓN EUROPEA

*"El FSE invierte en tu futuro"*

Fondo Social Europeo

  
Barrabés

 The Valley

# Sistemas Transaccionales y Bases de Datos

27 de Septiembre, 2023



Gobierno  
de España

VICERE<sup>SI</sup>NCIA  
PRIMERA DEL GOBIERNO  
MINISTERIO  
DE ASUNTOS ECONÓMICOS  
Y TRANSFORMACIÓN DIGITAL

SECRETARÍA DE ESTADO  
DE DIGITALIZACIÓN  
E INTELIGENCIA ARTIFICIAL

red.es

"El FSE invierte en tu futuro"

Fondo Social Europeo

Centro de  
Referencia Nacional  
en Comercio Electrónico  
y Marketing

CRN  
Digital



GARANTÍA  
JUVENIL



UNIÓN EUROPEA

 Barrabés

 The Valley

1. Ejercicio 1
2. SQL como DML
  - 2.1. Tabla
  - 2.2. Aserción
  - 2.3. Columnas restringidas
3. Agregaciones
  - 3.1. Funciones
  - 3.2. Group By
  - 3.3. Having
4. Insertar Datos
5. Ejercicio Final



# Ejercicio 1



# Ejercicio 1

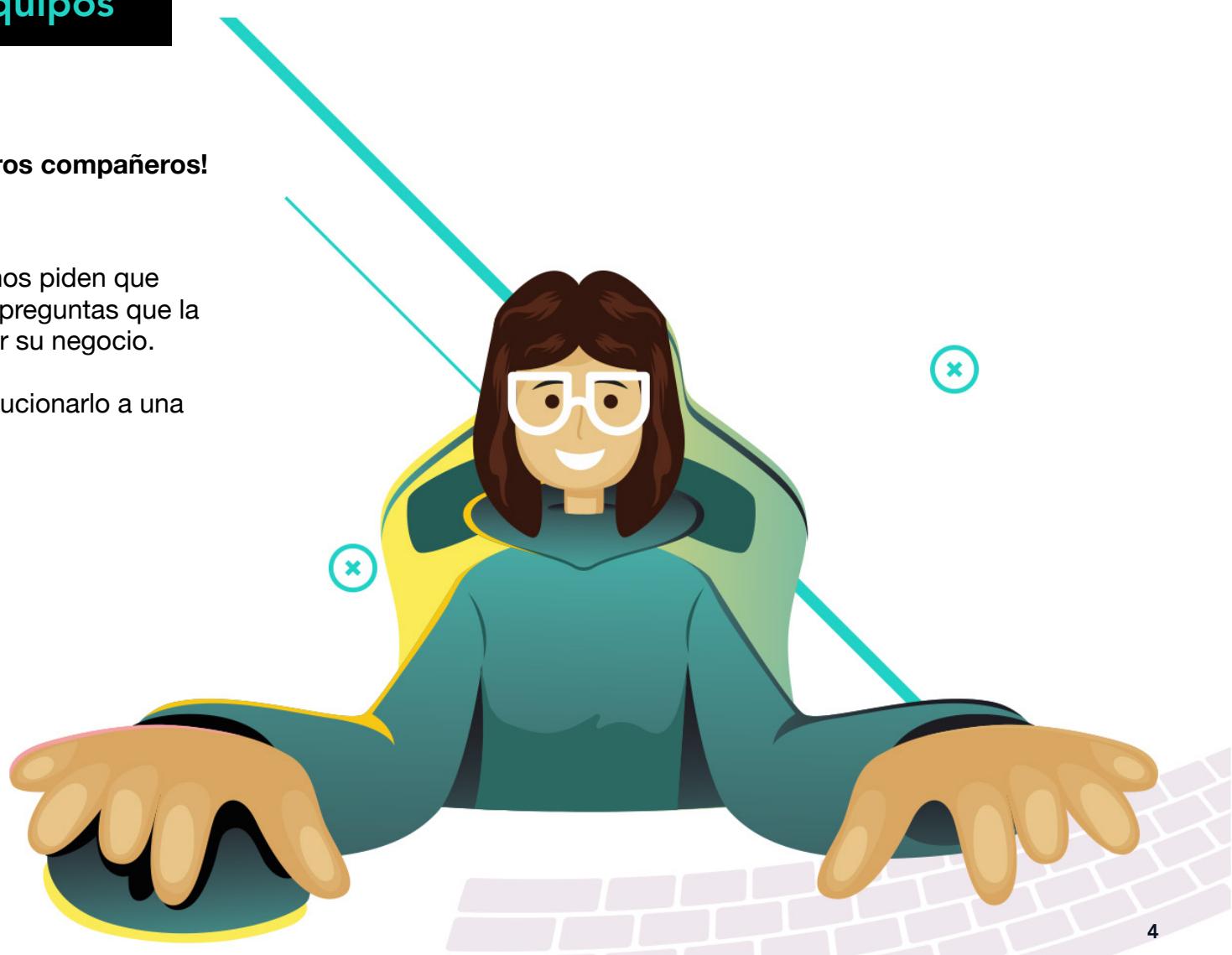
## Consultas a los datos de otros equipos

¡Vamos a trabajar con las bases de datos de nuestros compañeros!

Hemos llegado a un equipo nuevo, y en primer lugar, nos piden que estudiemos los datos y nos propongamos al menos 4 preguntas que la empresa necesitaría responder para comprender mejor su negocio.

Plantead las preguntas sin SQL, y luego tratad de evolucionarlo a una query SQL con lo que recordéis.

*Duración: 20 mins  
Metodología: Grupal*



# SQL - DML



# Queries de SQL como DML

## Crear una tabla

```
CREATE TABLE <nombre_tabla>
(<definicion_columna>,
[, <definicion_columna>, ...]
[, <restricciones_tabla>]
);
```



```
CREATE TABLE DEPARTAMENTOS
(
    nombre_dpt VARCHAR(20),
    ciudad_dpt VARCHAR(20),
    telefono VARCHAR(12) DEFAULT NULL,
    PRIMARY KEY (nombre_dpt, ciudad_dpt)
);
```

# Queries de SQL como DML

## Eliminar una tabla

**DROP TABLE <nombre\_tabla> {RESTRICT | CASCADE}**



**DROP TABLE Departamentos CASCADE**

# Queries de SQL como DML

## Crear una Aserción

**CREATE ASSERTION** <nombre\_asercion>  
**CHECK** (<condiciones>)



```
CREATE ASSERTION restriccion_ricos  
CHECK (NOT EXISTS (SELECT *  
         FROM proyectos p  
         LEFT JOIN empleados e  
         ON p.codigo_proy = e.num_proy  
         WHERE e.sueldo > 80000.0));
```

# Queries de SQL como DML

## Crear un filtro de valores

A veces nos interesa crear un filtro de posibles valores cuando creamos una tabla

Dependiendo de la plataforma en la que estemos, tendremos distintas formas de crear esta restricción.

**CREATE TABLE** Clientes

```
(  
    id CHAR(20) PRIMARY KEY,  
    nombre VARCHAR(50),  
    genero VARCHAR(50),  
    ciudad VARCHAR(20),  
    telefono VARCHAR(12) DEFAULT NULL,  
);
```

**CREATE TABLE** Clientes



```
(  
    id CHAR(20) PRIMARY KEY,  
    nombre VARCHAR(50),  
    genero CHECK( genero IN ('M','F','X') ),  
    ciudad VARCHAR(20),  
    telefono VARCHAR(12) DEFAULT NULL,  
);
```

Otras posibilidad es **ENUM**, la cual encontramos por ejemplo en MySQL.

No siempre tendremos la misma función para resolver un problema, pero podemos encontrar información al respecto si tenemos al menos una solución en otra plataforma

# Agregaciones



# Agregaciones

## Introducción

**Agregar datos** es útil, porque podemos **resumir información**, obteniendo **conocimientos** a partir de unos datos que se encuentran en estado bruto

GROUP BY agrupa las filas con los mismos valores en filas resumen

<b>COUNT(*)</b>	Counts observations in data table, takes NA in consideration
<b>COUNT(<i>field</i>)</b>	Counts non-NA observations in field
<b>COUNT(distinct <i>field</i>)</b>	Counts non-NA unique values in a field
<b>SUM(<i>field</i>)</b>	Returns the sum of all value (only numerical fields)
<b>MAX(<i>field</i>)</b>	Returns the largest value in a column
<b>MIN(<i>field</i>)</b>	Returns the smallest value in a column
<b>AVG(<i>field</i>)</b>	Returns the average value of a field (only numerical fields)

*Agregaciones Básicas en SQL*

# Agregaciones

**GROUP BY****SQL Query**

```
1 SELECT director_me, AVG(imdb_score)
2 FROM imdb_movies
3 GROUP BY 1;
```

**Output**

director_me	AVG(imdb_score)
	7.157894736842107
Adam Carolla	6.1
Adam Rifkin	6.2
Agustín Díaz Yanes	6.1
Akira Kurosawa	7.5
Al Franklin	4.3
Alan Yuen	6.2
Aleine Cal y Mayor	6.2
Alejandro G. Iñárritu	8.1
Aleksandr Veledinskiy	7.5
Alex Kendrick	6.9
Alex van Warmerdam	7
Alfonso Cuarón	7.9
Alfred Hitchcock	6.3

# Agregaciones

## GROUP BY evolucionado

¿Podemos organizar los valores para obtener las notas ordenadas?

Sí, con un **ORDER BY** [**<columna>**,...]

### SQL Query

```
1 SELECT director_me, AVG(imdb_score)
2 FROM imdb_movies
3 GROUP BY 1
4 ORDER BY avg(imdb_score) DESC;
```



### Output

director_me	AVG(imdb_score)
John Blanchard	9.5
John Stockwell	9.1
Mitchell Altieri	8.7
Cary Bell	8.7
Mike Mayhall	8.6
Charles Chaplin	8.6
Roger Allers	8.5
Raja Menon	8.5
Frank Darabont	8.5
S.S. Rajamouli	8.4
Catherine Owens	8.4
Sut Jhally	8.3

# Agregaciones

## GROUP BY evolucionado II

Si queremos meter un **WHERE**, tenemos que utilizar la función **HAVING**

### SQL Query

```
1 SELECT director_me, AVG(imdb_score) AS nota
2 FROM imdb_movies
3 GROUP BY 1
4 HAVING nota <5
5 ORDER BY avg(imdb_score) DESC;
```



director_me	nota
Nick Love	4.9
Jamel Debbouze	4.9
Isaac Florentine	4.9
H.M. Coakley	4.9
Vivek Agnihotri	4.8
Taedong Park	4.8
Sacha Bennett	4.8
Richard Schenkman	4.8
Neil Mcenery-West	4.8
Kevin Lima	4.8

# Agregaciones

## Consultas de Agregación a los Datos

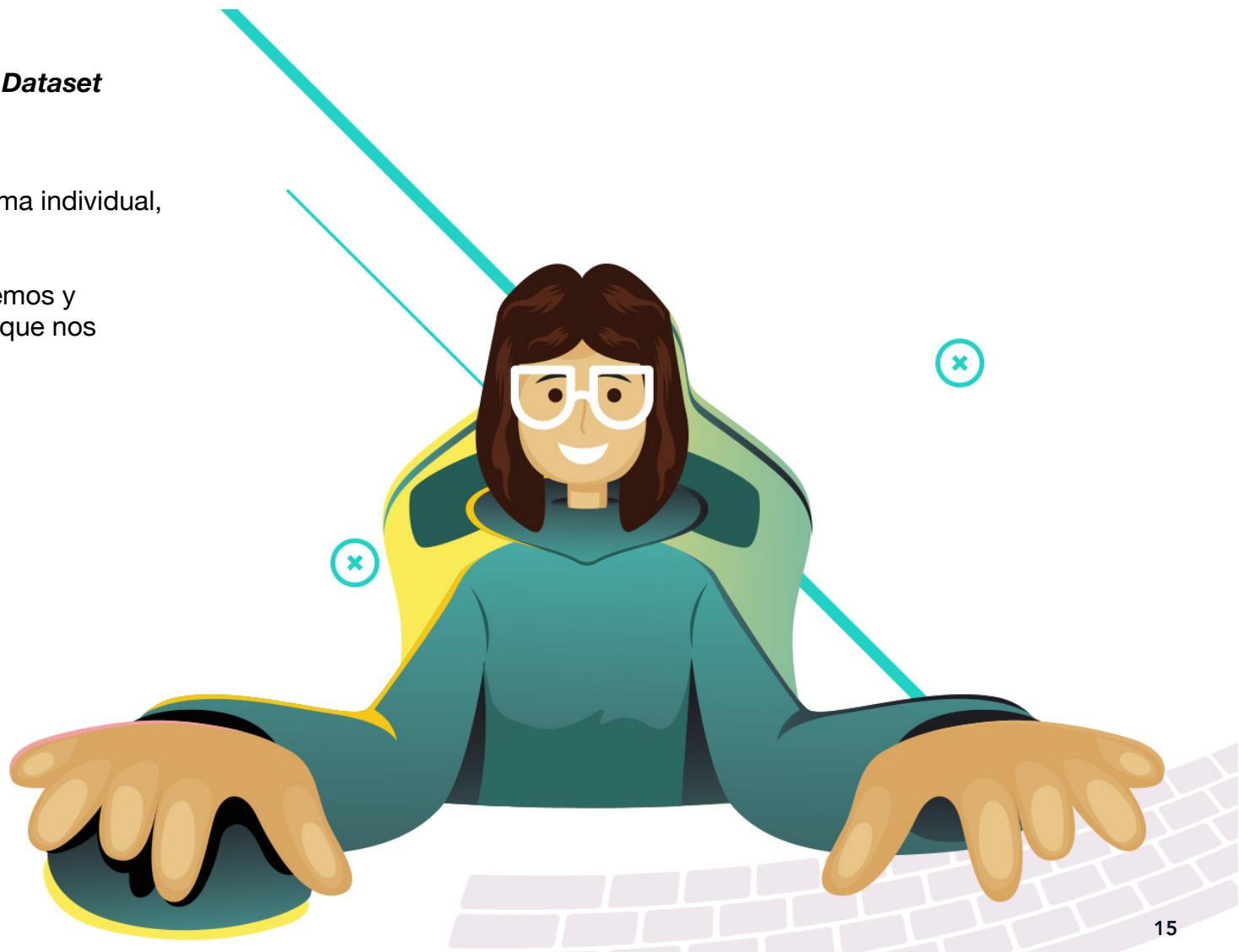
Trabajaremos con los datos de la carpeta **Dataset**

Vamos a seleccionar como mínimo dos dataset de forma individual, y nos los llevaremos a la plataforma de [SQLOnline](#).

Estudiamos los datos, las distintas columnas que tenemos y extraemos un mínimo de 4 consultas con GROUP BY que nos aporten información clave agregada de estas tablas.



Duración: 30 mins  
Metodología: Individual



# Insertar datos en Tablas



# Insertar datos en Tablas

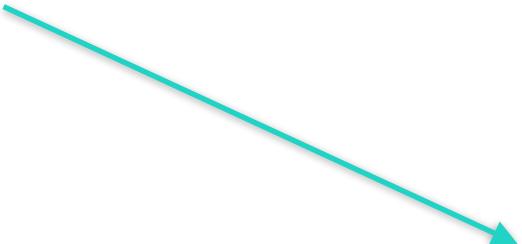
**INSERT**

¿Hemos pasado por alto una cosa al subir archivos a SQL Online!

Además de **CREATE TABLE**, hemos estado usando otra función adicional:

**INSERT INTO**

```
INSERT INTO table_name (column1, column2, column3, ...)  
VALUES (value1, value2, value3, ...);
```



```
CREATE TABLE 'Alumnos'  
('Nombre' TEXT, 'Telefono' INTEGER, 'Edad' INTEGER);
```

```
INSERT INTO 'Alumnos' ('Nombre', 'Telefono', 'Edad')  
VALUES ('Alex', '213454', '23'),  
        ('Juan', '2343546', '43'),  
        ('Inés', '2345', '23'),  
        ('Rosa', '12345', '14');
```

# Insertar datos en Tablas

## Creación y 'relleno' de tablas

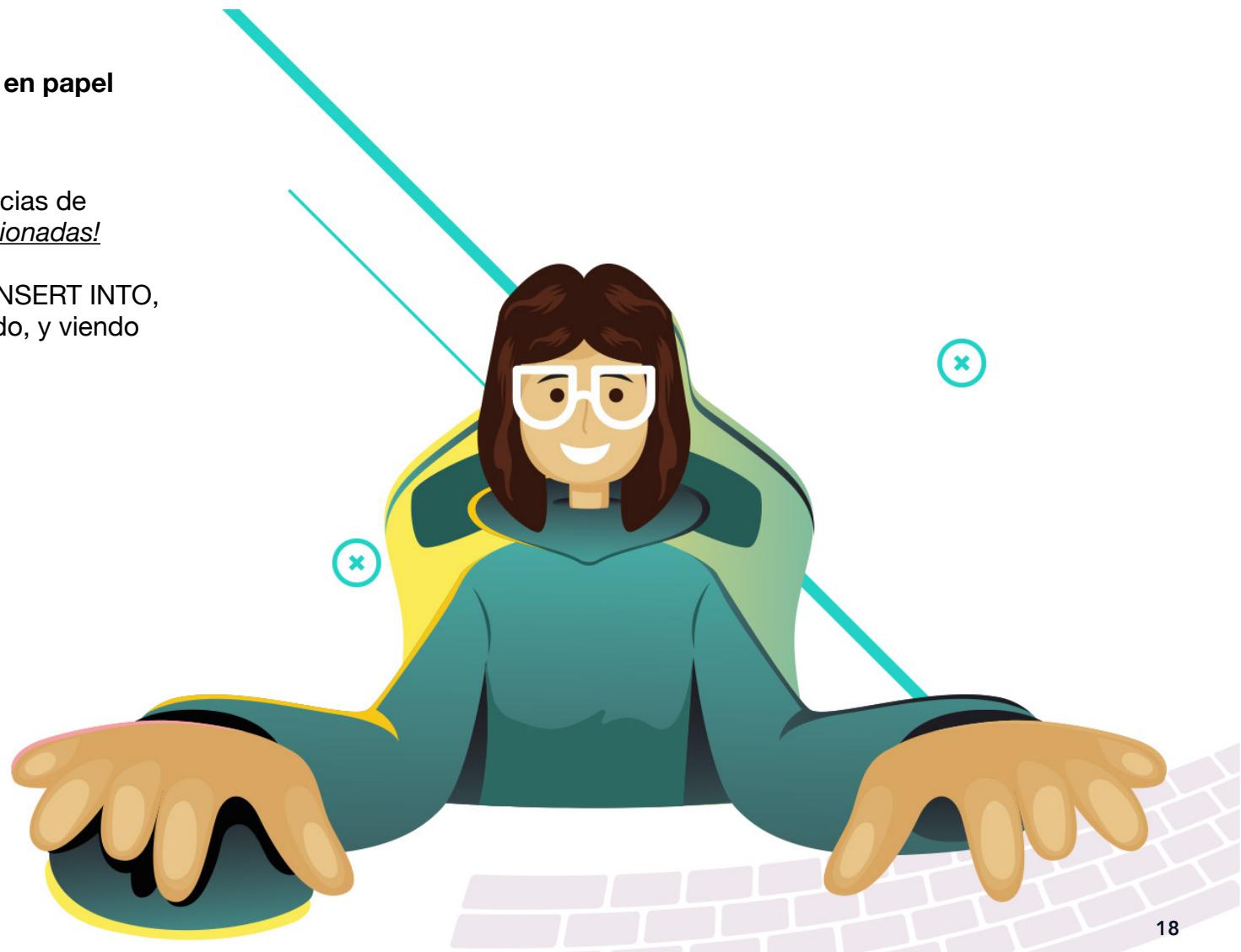
Habíamos creado una BBDD previamente, en papel

Vamos a crear como mínimo dos tablas con las instancias de CREATE TABLE. ¡Las tablas tienen que estar interrelacionadas!

Vamos también a probar a llenar las columnas con INSERT INTO, probando a meter los valores que habíamos predefinido, y viendo qué ocurre si introducimos uno repetido, no válido...



Duración: 20 mins  
Metodología: Individual



# Ejercicios Finales



# Práctica

## Ex.1: Descripción

Extraer campos *name*, *species* y *homeworld* de la tabla de datos de Star Wars

- *Tabla: star\_wars\_characters*

*Resultado esperado*

Row	name	species	homeworld
1	Bossk	Trandoshan	Trandosha
2	IG-88	Droid	<i>null</i>
3	R5-D4	Droid	Tatooine
4	R2-D2	Droid	Naboo
5	Nute Gunray	Neimodian	Cato Neimoidia
6	Mas Amedda	Chagrian	Champala
7	Adi Gallia	Tholothian	Coruscant
8	Mon Mothma	Human	Chandrila
9	Luke Skywalker	Human	Tatooine
10	Jek Tono Porkins	Human	Bestine IV

# Práctica

## Ex.2: Descripción

¿Cuáles son los planetas (*homeworlds*) incluídos en la tabla de Star Wars?

- Tabla: star\_wars\_characters

*Resultado esperado*

Row	homeworld
1	Trandosha
2	null
3	Tatooine
4	Naboo
5	Cato Neimoidia
6	Champala
7	Coruscant
8	Chandrila
9	Bestine IV
10	Eriadu

# Práctica

## Ex.3: Descripción

Extraer campos *film*, *director*, *year* y *actor* de la tabla de James Bond; filtrar por películas publicadas hasta el año 2000, cuyo director sea *Lewis Gilbert* o *Martin Campbell*. Excluir aquellas películas protagonizadas por Roger Moore.

- *Tabla: James Bond*

*Resultado esperado*

Row	Film	Director	Year	Actor
1	You Only Live Twice	Lewis Gilbert	1967	Sean Connery
2	GoldenEye	Martin Campbell	1995	Pierce Brosnan

# Práctica

## Ex.4: Descripción

Extraer países que cumplan con alguna de las siguientes condiciones: (i) sean países africanos con un índice de alfabetismo entre el 25% y 75% o (ii) países europeos con un ratio de población viviendo en áreas urbanas menor al 50%.

- Tabla: world\_health\_org

*Resultado esperado*

Row	Country	Continent	Adult_literacy_rate	Population_in_urban_areas
1	Uganda	Africa	68.1	13
2	Bosnia and Herzegovi	Europe	96.7	46
3	Turkmenistan	Europe	98.8	47
4	Sierra Leone	Africa	34.8	41
5	Niger	Africa	28.7	17
6	Sudan	Africa	60.9	42
7	Central African Republic	Africa	48.6	38
8	Liberia	Africa	60.0	59
9	Angola	Africa	67.4	54
10	Chad	Africa	25.7	26

## Práctica

### Ex.5: Descripción

¿Cuáles son los 5 países africanos con mayor PIB (GDP) per cápita?

- Tabla: world\_health\_org

*Resultado esperado*

Row	Country	Continent	Gross_income_per_capita
1	Equatorial Guinea	Africa	16620
2	Seychelles	Africa	14360
3	Botswana	Africa	11730
4	Gabon	Africa	11180
5	Mauritius	Africa	10640

# Práctica

## Ex.6: Descripción

Extraer las 10 películas con mayor *IMDB score*, filtrar por películas publicadas a partir de la década del 80, excluir aquellas producidas en los EEUU.

- Tabla: `imdb_movies`

## Resultado esperado

Row	movie_title	director_me	imdb_score
1	The Lord of the Rings: The Fellowship of the Ring	Peter Jackson	8.8
2	Queen of the Mountains	Sadyk Sher-Niyaz	8.7
3	City of God	Ferndo Meirelles	8.7
4	Spirited Away	Hayao Miyazaki	8.6
5	The Pianist	Roman Polanski	8.5
6	Children of Heaven	Majid Majidi	8.5
7	The Lives of Others	Florian Henckel von Donnersmarck	8.5
8	Airlift	Raja Menon	8.5
9	Amélie	Jean-Pierre Jeunet	8.4
10	Baahubali: The Beginning	S.S. Rajamouli	8.4

# Ejercicio Final Miércoles

## Consultas de Agregación a tablas creadas

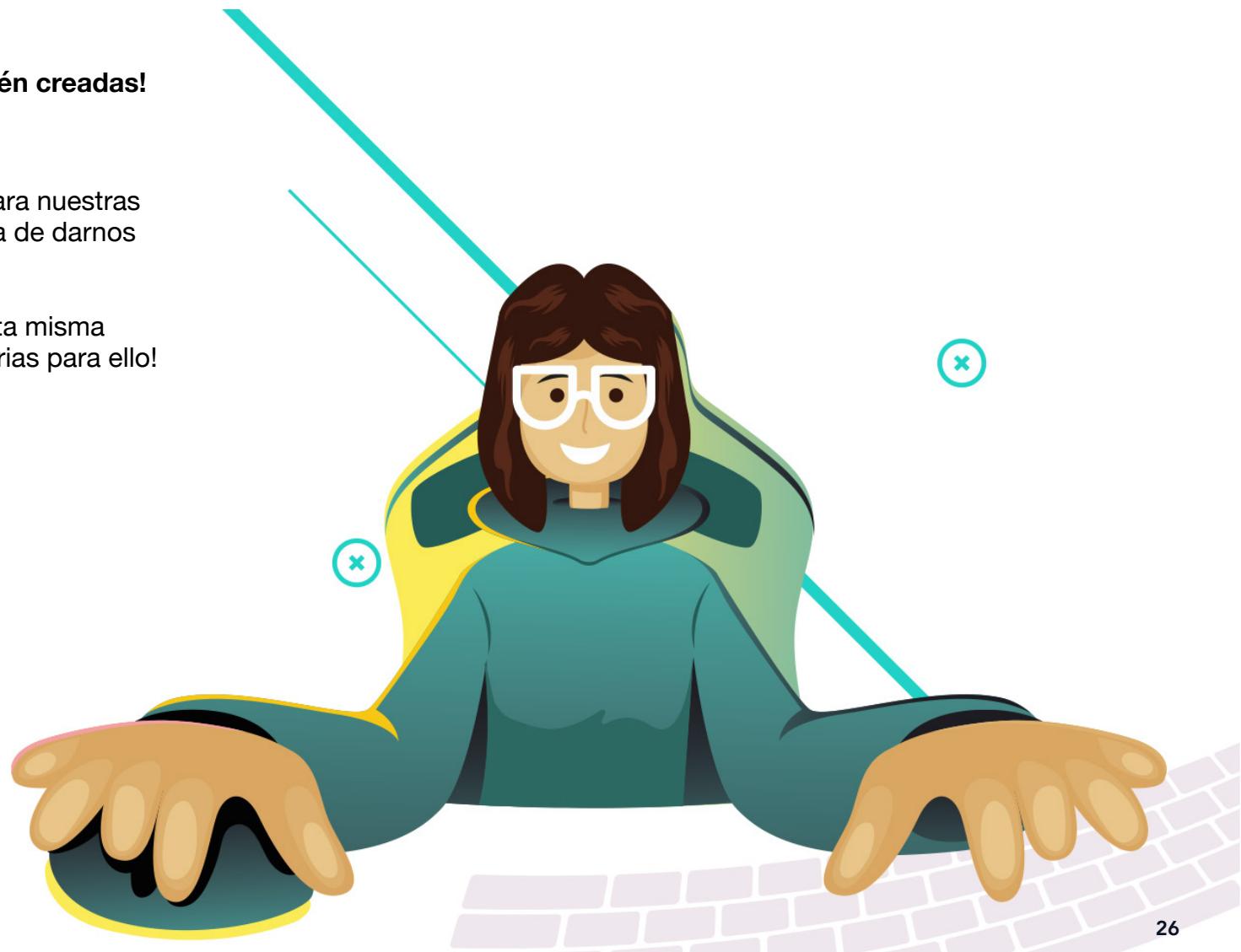
¡Vamos a crear consultas para las tablas recién creadas!

Esta misma mañana hemos desarrollado preguntas para nuestras tablas, por lo que crear consultas de nuevo no debería de darnos mucho problema.

Por supuesto, podremos reutilizar las consultas de esta misma mañana, ¡pero tendremos que tener las tablas necesarias para ello!



Duración: 20 mins  
Metodología: Individual





GOBIERNO  
DE ESPAÑA

VICEPRESIDENCIA  
PRIMERA DEL GOBIERNO  
  
MINISTERIO  
DE ASUNTOS ECONÓMICOS  
Y TRANSFORMACIÓN DIGITAL

SECRETARÍA DE ESTADO  
DE DIGITALIZACIÓN  
E INTELIGENCIA ARTIFICIAL

# red.es

Centro de  
Referencia Nacional  
en Comercio Electrónico  
y Marketing  
**CRN**  
Digital



UNIÓN EUROPEA

*“El FSE invierte en tu futuro”*

Fondo Social Europeo

  
Barrabés

 The Valley

# Sistemas Transaccionales y Bases de Datos

28 de Septiembre, 2023



"El FSE invierte en tu futuro"  
Fondo Social Europeo

1. Ejercicio Mañaneros
2. Case When
3. Joins
  - 3.1. Teoría de Conjuntos
  - 3.2. Tipos de Joins
4. Ejercicio Final



# Ejercicios Mañaneros



# Práctica

## Ex.1: Descripción

Extraer países que cumplan con alguna de las siguientes condiciones: (i) sean países africanos con un índice de alfabetismo entre el 25% y el 30% o (ii) países europeos con un ratio de población viviendo en áreas urbanas menor al 50%.

- Tabla: world\_health\_org
- Descripción de tabla: contiene información de países provista por la Organización Mundial de la Salud

Resultado esperado

Rank	Country	Continent	Adult_literacy_rate	Population_in_urban_areas
1	Cote d'Ivoire	Africa	68.1	13
2	Bosnia and Herzegovina	Europe	96.7	46
3	Turkmenistan	Europe	98.8	47
4	Sierra Leone	Africa	34.8	41
5	Niger	Africa	28.7	17
6	Sudan	Africa	60.9	42
7	Central African Republic	Africa	48.6	38
8	Liberia	Africa	60.0	59
9	Angola	Africa	67.4	54
10	Chad	Africa	25.7	26

## Práctica

### Ex.2: Descripción

¿Cuáles son los 5 países africanos con mayor PIB (GDP) per cápita?

- Tabla: world\_health\_org
- Descripción de tabla: contiene información de países provista por la Organización Mundial de la Salud

Resultado esperado

HECHO

Rank	Country	Continent	Gross_income_per_capita
1	Ecuadorian Guinea	Africa	16620
2	Seychelles	Africa	14360
3	Botswana	Africa	11730
4	Gabon	Africa	11180
5	Mauritius	Africa	10640

# Práctica

## Ex.3: Descripción

Extraer las 10 películas con mayor *IMDB score*, filtrar por películas publicadas a partir de la década del 80, excluir aquellas producidas en los EEUU.

- *Tabla:* imdb\_movies
- *Descripción de la tabla:* esta tabla incluye data de películas publicada por IMDB (imdb.com)

Resultado esperado

HECHO

Row	Movie_title	director_me	imdb_score
1	The Lord of the Rings: The Fellowship of the Ring	Peter Jackson	8.8
2	The Return of the King	Sadyk Sher-Niyaz	8.7
3	City of God	Fernando Meirelles	8.7
4	Spirited Away	Hayao Miyazaki	8.6
5	The Pianist	Roman Polanski	8.5
6	Children of Heaven	Majid Majidi	8.5
7	The Lives of Others	Florian Henckel von Donnersmarck	8.5
8	Airlift	Raja Menon	8.5
9	Amélie	Jean-Pierre Jeunet	8.4
10	Baahubali: The Beginning	S.S. Rajamouli	8.4

# Case When



# Case When Statement

## Definiciones en esta sección

El **CASE WHEN** statement permite crear campos con condiciones específicas, donde el valor que adquiera cada fila dependerá del criterio establecido. El statement se invoca utilizando la siguiente sintaxis,

**CASE**

**WHEN** condition\_1 **THEN** value\_if\_true

**WHEN** condition\_2 **THEN** value\_if\_true

**ELSE** value\_if\_none\_above\_is\_true

**END AS** field\_name

# Case When Statement

## Ejemplo utilizando un CASE WHEN statement

### SQL Query

```

1 SELECT
2     Company,
3     Revenue,
4     CASE
5         WHEN Revenue > 200000 THEN 'Top Revenue Company'
6         WHEN Revenue > 100000 THEN 'Mid Revenue Company'
7         ELSE 'Not in Top'
8     END AS 'Revenue Segment'
9 FROM fortune
10 ORDER BY Revenue DESC

```

### Output

Company	Revenue	revenue segment
Walmart	482130	top revenue company
Exxon Mobil	246204	top revenue company
Apple	233715	top revenue company
Berkshire Hathaway	210821	top revenue company
Chevron	131118	mid revenue company
Costco	116199	mid revenue company
Fannie Mae	110359	mid revenue company
Amazon.com	107006	mid revenue company
HP	103355	mid revenue company
J.P. Morgan Chase	101006	mid revenue company
Microsoft	93580	not in the top

# Joins



# Joins

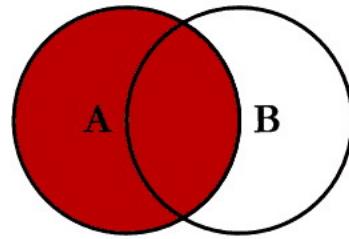
## Teoría de Conjuntos

¿Qué sabéis de cruzar cosas?

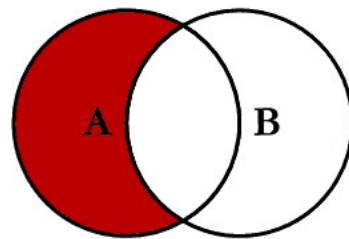
# Joins

## Tipos de Joins

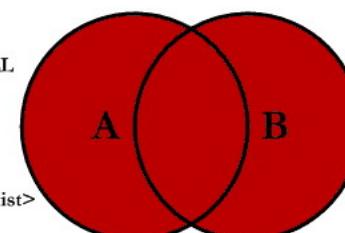
## SQL JOINS



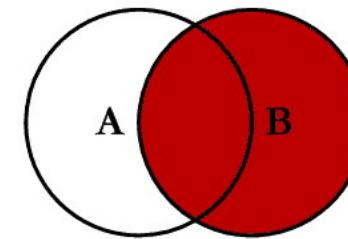
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



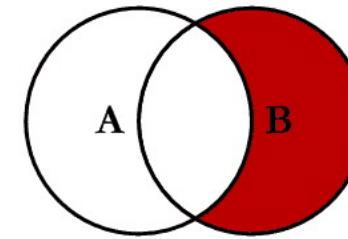
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

© C.L. Moffatt, 2008

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

# Joins

## Ejercicio Inicial Joins

¡Vamos a trabajar con las tablas que creamos ayer!

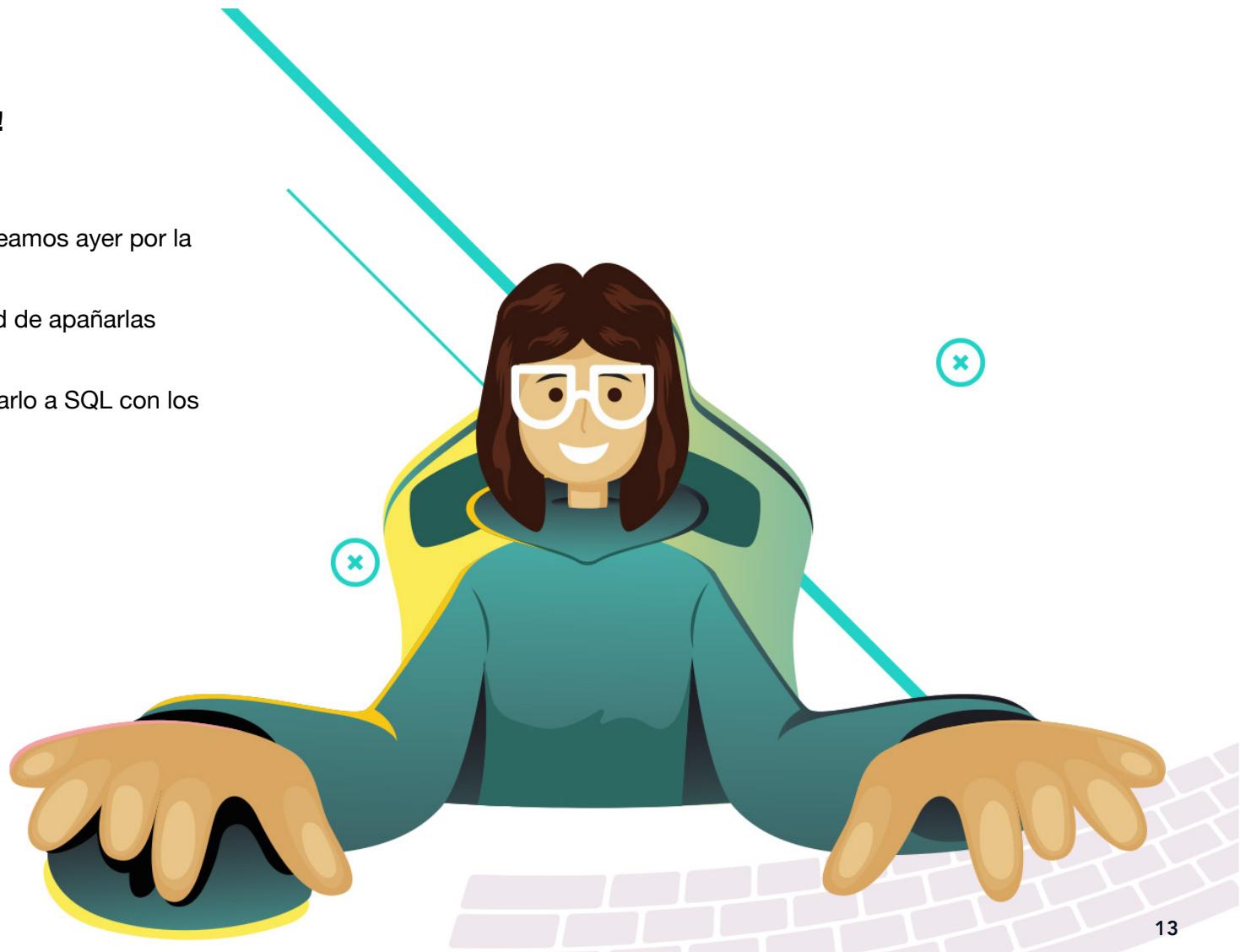
Vamos a realizar al menos un join con las dos tablas que creamos ayer por la tarde.

Recuperad el código de ayer, creadlas de nuevo (o terminad de apañarlas sobre la marcha) ¡y a currar!

Plantead las preguntas sin SQL y luego tratad de evolucionarlo a SQL con los procedimientos comentados.

Opcional: exponerlo y explicarlo sobre la marcha

Duración: 15 mins  
Metodología: Grupal



# Joins

## Ejercicio Final

¡Busquemos datos más complejos!

Vamos a utilizar Kaggle para trabajar con una BBDD más avanzada (y con más datos).

Someteremos a votación cuál utilizar, y nos lo traeremos a SQLOnline.

Plantead las posibles preguntas a esta nueva BBDD, y realizad las consultas SQL.

kaggle



Duración: 40 mins  
Metodología: Grupal



# Joins

## Ejercicio Final V<sub>2</sub>

¡Vamos a trabajar con las bases de datos de nuestros compañeros!

Ya llevamos un rato trabajando con las BBDD de nuestros nuevos equipos, tenemos que realizar consultas más avanzadas

Plantead las preguntas sin SQL, el recorrido a seguir a lo largo del DataSet y luego tratad de evolucionarlo a SQL con los procedimientos comentados.

Opcional: exponerlo y explicarlo sobre la marcha

Duración: 30 mins  
Metodología: Grupal



# Ejercicios Jueves



## Ex.1: Descripción

Extraer consolas no discontinuadas y su fecha de lanzamiento (*first retail availability*). Agregar ventas globales de videojuegos publicados desde el año 2000. Ordenar según ventas totales de videojuegos (descendiente).

- *Table 1: videogames\_games*
- *Table 2: videogames\_consoles*

*Resultado esperado*

Row	console_name	first_retail_availability	global_games_sales
1	Playstation 3	2006-11-11	945.0
2	Nintendo 3DS	2011-02-26	240.0
3	Platstation 4	2013-11-15	239.0
4	Xbox One	2013-11-22	129.0
5	Nintendo WiiU	2012-11-18	79.0
6	Play Station Vita	2011-12-17	58.0

## Ex.2: Descripción

Los directores de películas de James Bond han trabajado en promedio en dos películas cada uno. ¿Cómo obtendrías esta información?

Tabla: **james\_bond**

Resultado esperado

Row	DIR_AVG_MOVIES
1	2.0

## Ex.3: Descripción

Agregar las siguientes métricas para todos los países africanos,

- average gross income per capita
- total population
- number of countries

Tabla: world\_health\_org

Descripción de la tabla: esta tabla contiene data de países del mundo publicados por la Organización Mundial de la Salud

Resultado esperado

Row	africa_avg_GDP	total_population	count_countries
1	3127.95	759147	46

## Ex.4: Descripción

Calcular número de personajes según planeta (*homeworld*). Evitar personajes sin información sobre planeta de origen.

Tabla: ***star\_wars\_characters***

Descripción de la tabla: esta tabla incluye datos sobre personajes de Star Wars

Resultado esperado

Row	homeworld	count_characters
1	Naboo	11
2	Tatooine	10
3	Coruscant	3
4	Kamino	3
5	Alderaan	3

## Ex.5: Descripción

Calcular el total de salario percibido por cada actor en todas las películas. Omitir películas sin data sobre salario.

*Tabla: james\_bond*

*Descripción de la tabla:* esta tabla reporta data sobre películas de James Bond

*Resultado esperado*

Row	Actor	bond_salary
1	Pierce Brosnan	46.5
2	Daniel Craig	25.9
3	Sean Connery	20.3
4	Roger Moore	16.9
5	Timothy Dalton	13.1
6	George Lazenby	0.6

## Ex.6: Descripción

¿Podemos asegurar que las películas de acción tienen de media mejor valoración que el resto de películas? Extraer total de películas y media de IMDB score para películas de acción vs. el resto (de forma conjunta).

Tabla: *imdb\_movies*

Resultado esperado

gender	total_movies	AVG_imdb_score
Action Movies	84	6.52
Rest of Movies	416	6.52

## Ex.7: Descripción

Calcular la facturación (*box office*) según director. Filtrar por aquellos directores que hayan generado más de 1500 en el total de facturación (todas las películas).

*Tabla: james\_bond*

*Descripción de la tabla:* esta tabla reporta data sobre películas de James Bond

*Resultado esperado*

Row	Director	total_box_office
1	Guy Hamilton	2057.0
2	Terence Young	1841.0
3	Sam Mendes	1670.0
4	John Glen	1663.0
5	Lewis Gilbert	1582.0

## Ex.8: Descripción

Calcular número total de álbumes según *sub metal genre*, filtrar por aquellos subgéneros con al menos 10 álbumes.

Tabla: *rolling\_top\_albums*

Descripción de la tabla: esta tabla incluye datos sobre álbumes de música heavy metal

Resultado esperado

Row	Sub_Metal_Genre	count_albums
1	Thrash Metal	15
2	Heavy Metal	14
3	Alternative Metal	10

## Ex.9: Descripción

¿Cuántos artistas hay incluídos en el dataset cuyo nombre incluye las palabras ‘god’, ‘death’ or ‘black’?

Tabla: ***rolling\_top\_albums***

Descripción de la tabla: esta tabla incluye datos sobre álbumes de música heavy metal

Resultado esperado

Row	artist_keyword	count_artist
1	GOD	3
2	DEATH	2
3	BLACK	1



GOBIERNO  
DE ESPAÑA

VICEPRESIDENCIA  
PRIMERA DEL GOBIERNO  
  
MINISTERIO  
DE ASUNTOS ECONÓMICOS  
Y TRANSFORMACIÓN DIGITAL

SECRETARÍA DE ESTADO  
DE DIGITALIZACIÓN  
E INTELIGENCIA ARTIFICIAL

red.es

Centro de  
Referencia Nacional  
en Comercio Electrónico  
y Marketing

CRN  
Digital



UNIÓN EUROPEA

*"El FSE invierte en tu futuro"*

Fondo Social Europeo

  
Barrabés

 The Valley

# Sistemas Transaccionales y Bases de Datos

29 de Septiembre, 2023



red.es



"El FSE invierte en tu futuro"  
Fondo Social Europeo

1. Subqueries
2. Fechas
3. Actualizar Tablas
4. Ejercicios Guiados
5. Ejercicio Final



# Subqueries



# Subqueries

## ¿Introducción?

Guardamos una **tabla temporal** en memoria para poder trabajar con ella como si estuviera en nuestro esquema

**WITH**

```
temp_table_1 AS (SELECT... FROM...),  
temp_table_2 AS (SELECT... FROM...)  
SELECT... FROM temp_table_1
```



### SQL Query

```
1 WITH my_subquery AS (SELECT Sector,  
2                               SUM(Revenue) AS total_revenue  
3                         FROM fortune  
4                           GROUP BY 1)  
5 SELECT AVG (total_revenue) AS avg_sector_revenue  
6 FROM my_subquery
```

### Output

avg\_sector\_revenue

876000

# Unions



# Unions

## UNION ALL

Una UNION combina dos tablas de forma vertical (uniendo filas).  
**UNION ALL incluye todas las filas, sin importar si son duplicados.**

```
SELECT column1, column2  
FROM table1  
UNION ALL  
SELECT column1, column2  
FROM table2
```



Table1	
column1	column2
a	b
a	c
a	d

Union-ALL

U

Table 2	
column1	column2
b	c
a	d

=

Table1 Union Table2	
column1	column2
a	b
a	c
b	c
a	d
a	d

Duplicate Rows  
are Repeated in  
Results

# Unions

## UNION DISTINCT

Una UNION combina dos tablas de forma vertical (uniendo filas).  
**UNION DISTINCT excluye los duplicados del resultado final**

```
SELECT column1, column2  
FROM table1  
UNION DISTINCT  
SELECT column1, column2  
FROM table2
```



Table1	
column1	column2
a	b
a	c
a	d

Union-Distinct

U

Table 2	
column1	column2
b	c
a	d

=

Table1 Union Table2	
column1	column2
a	b
a	c
a	d
b	c

Duplicate row  
not repeated in  
results

# Ejercicio UNIONS

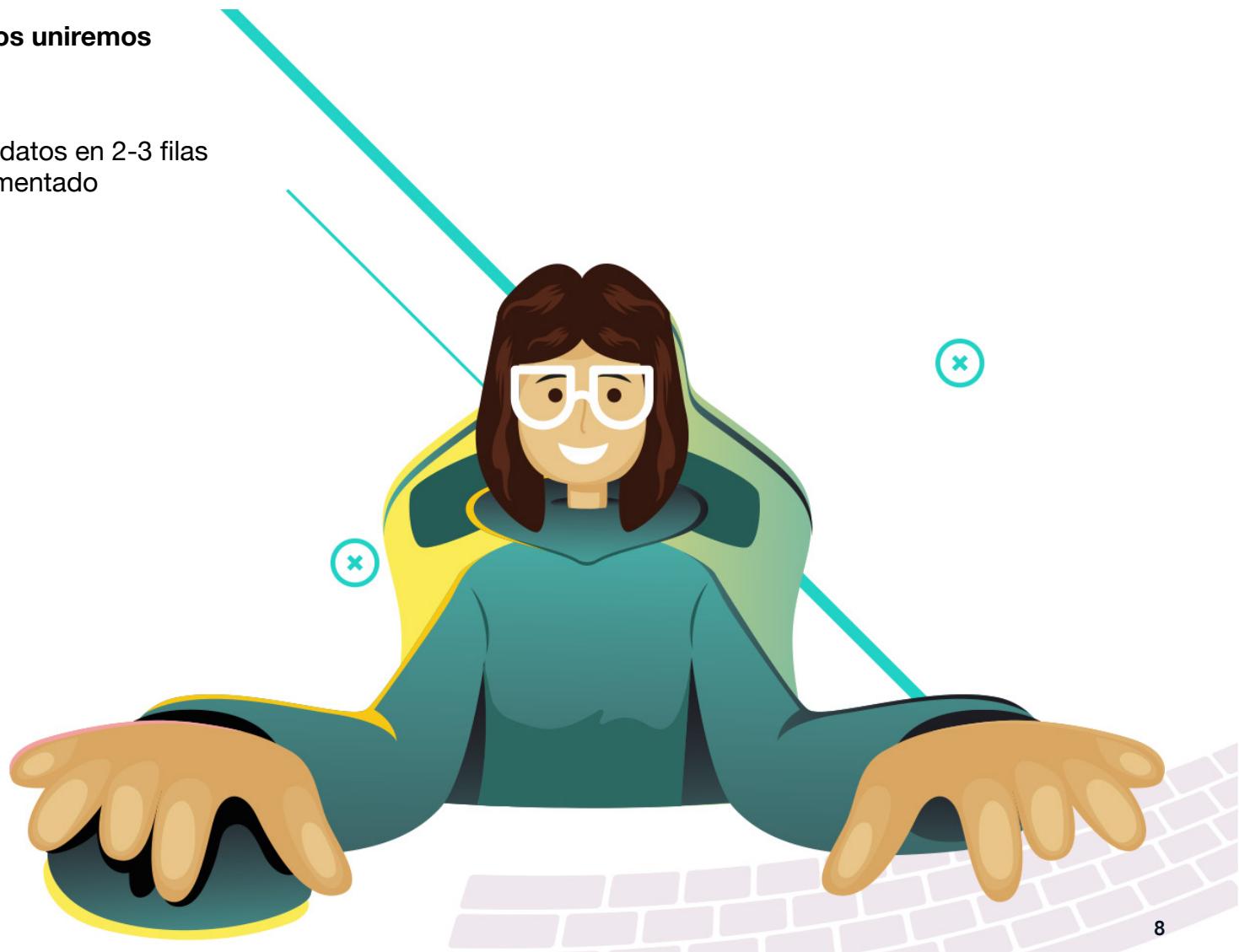
## Ejemplo sencillo

Crearemos un ejemplo sencillo de dos csv y los uniremos

Preparar dos tablas en Excel con formato .csv, añadir datos en 2-3 filas y comprobar los dos tipos de UNIONES que hemos comentado

- UNION (UNION ALL)
- UNION DISTINCT

*Duración: 15 mins  
Metodología: Individual*



# Fechas en SQL



# Fechas en SQL

## EXTRACT y DATE\_TRUNC

Conocéis las funciones de **EXTRACT** y **DATE\_TRUNC**

SELECT

```
Date,  
EXTRACT(DAY FROM Date) as day_number,  
EXTRACT(WEEK FROM Date) as week_number ,  
EXTRACT(MONTH FROM Date) as month_number ,  
EXTRACT(YEAR FROM Date) as year_number ,  
DATE_TRUNC(Date,WEEK) as week_starting_date ,  
DATE_TRUNC(Date,MONTH) as month_starting_date
```

```
FROM amazon  
ORDER BY Date
```

### Date Parts

- **year = year**
- **quarter = quarter**
- **month = month**
- **dayofyear = day of the year**
- **dayofweek = day of the week**
- **day = day of the month**
- **week = week**
- **hour = hour**
- **time = time**
- **minute = minute**
- **second = second**
- **millisecond = millisecond**

# Fechas en SQL

## EXTRACT y DATE\_TRUNC

Row	Date	day_number	week_number	month_number	year_number	week_starting_date	week_starting_date_1
1	2010-01-04	4	1	1	2010	2010-01-03	2010-01-01
2	2010-01-05	5	1	1	2010	2010-01-03	2010-01-01
3	2010-01-06	6	1	1	2010	2010-01-03	2010-01-01
4	2010-01-07	7	1	1	2010	2010-01-03	2010-01-01
5	2010-01-08	8	1	1	2010	2010-01-03	2010-01-01
6	2010-01-11	11	2	1	2010	2010-01-10	2010-01-01
7	2010-01-12	12	2	1	2010	2010-01-10	2010-01-01
8	2010-01-13	13	2	1	2010	2010-01-10	2010-01-01
9	2010-01-14	14	2	1	2010	2010-01-10	2010-01-01
10	2010-01-15	15	2	1	2010	2010-01-10	2010-01-01

# Ejercicio Fechas

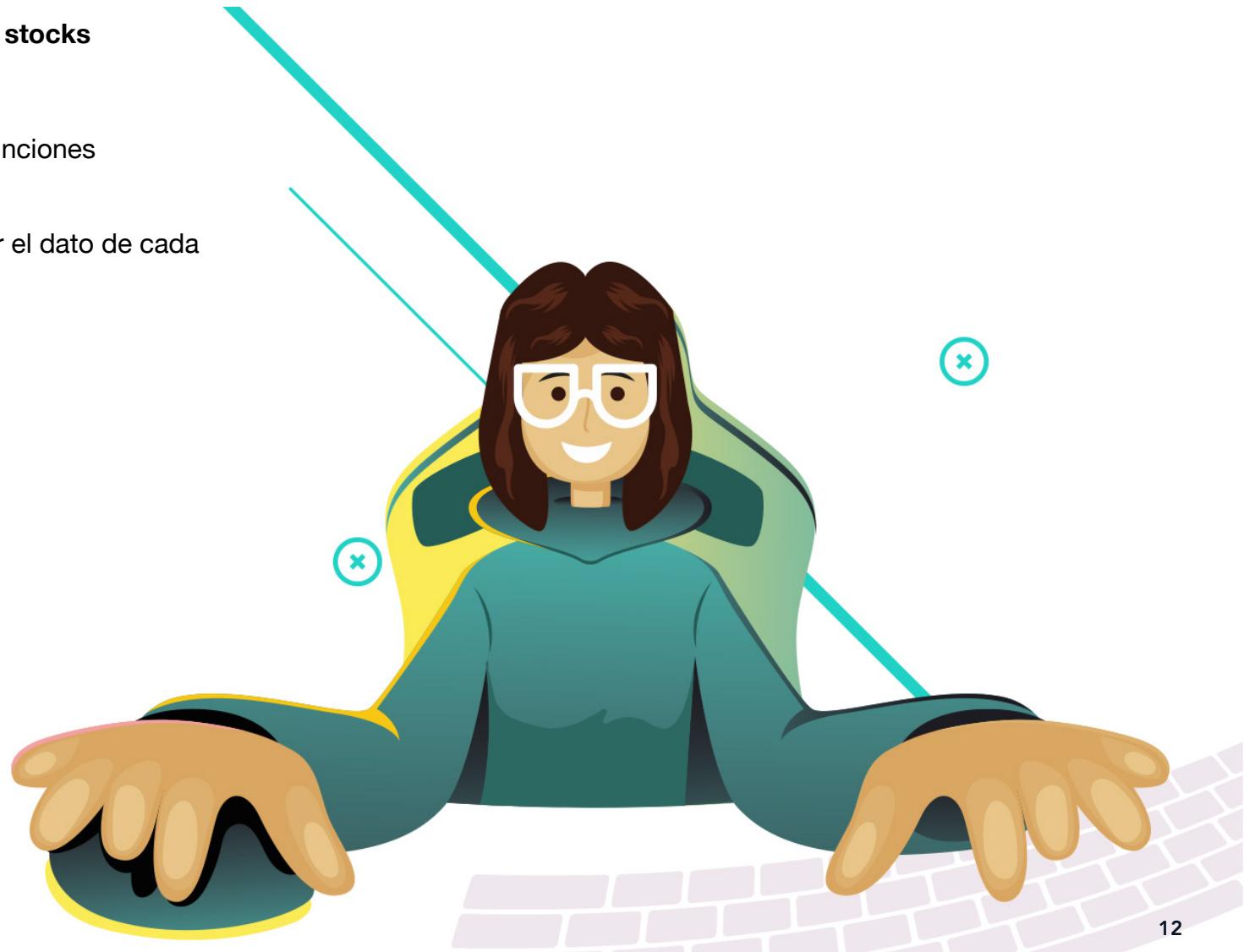
## Ejemplo sencillo

Vamos a coger la tabla de Amazon trade stocks

Comprobar si funciona en SQLite alguna de las dos funciones comentadas.(spoiler: no funciona)

Tratar de conseguir el código que nos permita obtener el dato de cada fecha desglosado.

Duración: 15 mins  
Metodología: Grupal



# Actualizar Tablas



# Actualizar Tablas

## Update

Para añadir un registro nuevo a nuestra tabla, hemos estado utilizando **INSERT INTO**, pero ¿y si queremos modificar algún registro que teníamos previamente?  
¡Podemos utilizar **UPDATE**!

```
UPDATE table_name  
SET column1 = value1,  
    column2 = value2....,  
    columnN = valueN  
WHERE [condition];
```



```
UPDATE Empleados  
SET salario = 3000.00,  
    convenio = 'CV0921'  
WHERE idemp = 'X042321';
```

## Replace

```
REPLACE(columna, 'valor_inicial', 'valor_final')
```



```
REPLACE(salario, ',', '')
```



3,000.00 pasará a 3000.00

# MongoDB



# MongoDB

¿Qué es MongoDB?

!!!!¿NO TODO ES SQL?!!!!



Es una base de Datos distribuida, gratis y de código abierto

Los documentos son almacenados en una estructura parecida a un JSON (BSON)

```
{  
    "name": "Carlos González",  
    "age": 25,  
    "city": "Alicante",  
    "cars": [  
        {  
            "model": "Renault",  
            "year": 2022,  
            "value": 20000  
        },  
        {  
            "model": "Chevrolet"  
            "year": 2009  
            "value": 10000  
        }  
    ]  
}
```

# MongoDB

## Conceptos Básicos

### Estructuras Principales de MongoDB

#### Documentos

Un registro dentro de una colección  
Es análogo a un objeto JSON (BSON)  
La unidad básica dentro de MongoDB

#### Colecciones

Agrupación de documentos  
Equivalente a una tabla en las bases de datos relacionales  
No impone un esquema

#### Base de datos

Contenedor físico de colecciones  
Cada base de datos tiene su archivo propio en el sistema de archivos  
Un cluster puede tener múltiples bases de datos

# MongoDB

## Conceptos Resumen

1. MONGO DB ES UNA **BBDD DISTRIBUIDA**, LO QUE NOS PERMITE **ESCALAR DE FORMA HORIZONTAL**.
2. EL **URI** (UNIFORM RESOURCE IDENTIFIER, “IDENTIFICADOR UNIFORME DE RECURSOS”) **IDENTIFICA UN RECURSO** POR SU NOMBRE, POR SU UBICACIÓN O POR AMBOS. EN ESTE ÚLTIMO CASO, EL URI INDICA QUE UN RECURSO IDENTIFICADO Y DÓNDE ESTÁ DISPONIBLE. COMPRENDE EL URL Y EL URN. **NO DEBE ESTAR EN NUESTRO REPOSITORIO PÚBLICO**
3. ES **BUENO ENcriptar DATOS GUARDADOS** EN NUESTRA BBDD/CLUSTER
4. LA FUNCIÓN **PRETTY** MUESTRA EL RESULTADO DE UNA **QUERY** EN CONSOLA DE MANERA **MÁS AMIGABLE**
5. EN MONGODB SE PUEDEN GUARDAR **ARREGLOS DE DOCUMENTOS** Y ES UNA FORMA DE **EXPRESAR RELACIONES DE UNO A MUCHOS**.
6. SU LECTURA NO ES TAN COMPLICADA DE ENTENDER, POR EJEMPLO SI EJECUTO **DB.COLLECTION.FIND()** NOS RETORNARÁ TODOS LOS DOCUMENTOS DE LA COLECCIÓN ‘COLLECTION’
7. SI QUIERO **INSERTAR MUCHOS DOCUMENTOS** EN LA BBDD, LO PODRÍAMOS HACER CON ‘**INSERT MANY**’
8. RECUERDA QUE **MONGODB ES SCHEMA LESS** (SIN ESQUEMA)
9. EN UN CLUSTER DE MONGODB PUEDE HABER **MUCHAS BASES DE DATOS**
10. LOS DOCUMENTOS ALMACENADOS EN MONGODB NO ES NECESARIO QUE TENGAN UNA MISMA ESTRUCTURA, DE HECHO ESTA ES UNA DE SUS PRINCIPALES VENTAJAS QUE HACE QUE **MONGODB SEA MÁS ESCALABLE Y FLEXIBLE**

# Ejercicios Guiados



# Ejercicios Guiados

¡Tod@s a una!

Vamos a utilizar varias de las tablas que encontramos en *Dataset*

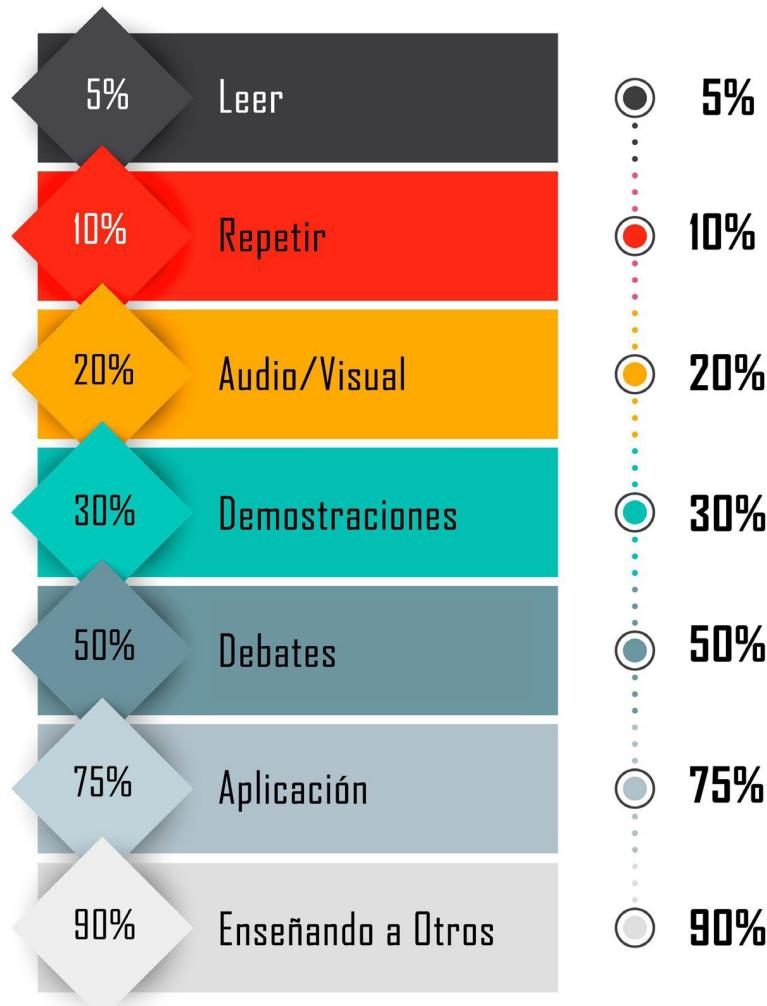
Propondremos una query para cada una de las tablas.\*

Duración: ¿? mins  
Metodología: Grupal



## ¿Por qué tanta interacción?

### La Pirámide del Aprendizaje





GOBIERNO  
DE ESPAÑA

VICEPRESIDENCIA  
PRIMERA DEL GOBIERNO

MINISTERIO  
DE ASUNTOS ECONÓMICOS  
Y TRANSFORMACIÓN DIGITAL

SECRETARÍA DE ESTADO  
DE DIGITALIZACIÓN  
E INTELIGENCIA ARTIFICIAL

red.es

Centro de  
Referencia Nacional  
en Comercio Electrónico  
y Marketing  
**CRN**  
Digital



UNIÓN EUROPEA

*“El FSE invierte en tu futuro”*

---

Fondo Social Europeo

  
**Barrabés**

 **The Valley**

# Fechas en SQLite

## strftime() y date()

¡El código equivalente en SQLite sería el siguiente!

**SELECT**

```
Date,
strftime('%d', Date) as day_number,
strftime('%W', Date) as week_number ,
strftime('%m', Date) as month_number ,
strftime('%Y', Date) as year_number,
date(Date, '-6 days', 'weekday 1') as week_starting_date,
date(Date, 'start of month') as month_starting_date
```

**FROM** amazon

**ORDER BY** date

## Ejercicio 2.0

Vamos a probar a utilizarlo en las tablas que tenemos de nuestras BBDD

Format	Description
%d	day of the month: 01-31
%f	fractional seconds: SS.SSS
%H	hour: 00-24
%j	day of the year: 001-366
%J	Julian day number
%m	month: 01-12
%M	minute: 00-59
%s	seconds since 1970-01-01
%S	seconds: 00-59
%w	day of week 0-6 with Sunday==0
%W	week of the year: 00-53
%Y	year: 0000-9999
%%	%

# Sistemas Transaccionales y Bases de Datos

29 de Septiembre, 2023



"El FSE invierte en tu futuro"  
Fondo Social Europeo

1. Subqueries
2. Fechas
3. Actualizar Tablas
4. Ejercicios Guiados
5. Ejercicio Final



# Subqueries



# Subqueries

## ¿Introducción?

Guardamos una **tabla temporal** en memoria para poder trabajar con ella como si estuviera en nuestro esquema

**WITH**

```
temp_table_1 AS (SELECT... FROM...),  
temp_table_2 AS (SELECT... FROM...)  
SELECT... FROM temp_table_1
```



### SQL Query

```
1 WITH my_subquery AS (SELECT Sector,  
2                               SUM(Revenue) AS total_revenue  
3                         FROM fortune  
4                           GROUP BY 1)  
5 SELECT AVG (total_revenue) AS avg_sector_revenue  
6 FROM my_subquery
```

### Output

avg\_sector\_revenue

876000

# Unions



# Unions

## UNION ALL

Una UNION combina dos tablas de forma vertical (uniendo filas).  
**UNION ALL incluye todas las filas, sin importar si son duplicados.**

**SELECT column1, column2**

**FROM table1**

**UNION ALL**

**SELECT column1, column2**

**FROM table2**



Table1	
column1	column2
a	b
a	c
a	d

**Union-ALL**

U =

Table 2	
column1	column2
b	c
a	d

Table1 Union Table2	
column1	column2
a	b
a	c
b	c
a	d
a	d

Duplicate Rows  
are Repeated in  
Results

# Unions

## UNION DISTINCT

Una UNION combina dos tablas de forma vertical (uniendo filas).  
**UNION DISTINCT excluye los duplicados del resultado final**

```
SELECT column1, column2  
FROM table1  
UNION DISTINCT  
SELECT column1, column2  
FROM table2
```



Table1	
column1	column2
a	b
a	c
a	d

Union-Distinct

U

Table 2	
column1	column2
b	c
a	d

=

Table1 Union Table2	
column1	column2
a	b
a	c
a	d
b	c

Duplicate row  
not repeated in  
results

# Ejercicio UNIONS

## Ejemplo sencillo

Crearemos un ejemplo sencillo de dos csv y los uniremos

Preparar dos tablas en Excel con formato .csv, añadir datos en 2-3 filas y comprobar los dos tipos de UNIONES que hemos comentado

- UNION (UNION ALL)
- UNION DISTINCT

*Duración: 15 mins  
Metodología: Individual*



# Fechas en SQL



# Fechas en SQL

## EXTRACT y DATE\_TRUNC

Conocéis las funciones de **EXTRACT** y **DATE\_TRUNC**

SELECT

```
Date,  
EXTRACT(DAY FROM Date) as day_number,  
EXTRACT(WEEK FROM Date) as week_number ,  
EXTRACT(MONTH FROM Date) as month_number ,  
EXTRACT(YEAR FROM Date) as year_number ,  
DATE_TRUNC(Date,WEEK) as week_starting_date ,  
DATE_TRUNC(Date,MONTH) as month_starting_date
```

```
FROM amazon  
ORDER BY Date
```

### Date Parts

- **year = year**
- **quarter = quarter**
- **month = month**
- **dayofyear = day of the year**
- **dayofweek = day of the week**
- **day = day of the month**
- **week = week**
- **hour = hour**
- **time = time**
- **minute = minute**
- **second = second**
- **millisecond = millisecond**

# Fechas en SQL

## EXTRACT y DATE\_TRUNC

Row	Date	day_number	week_number	month_number	year_number	week_starting_date	week_starting_date_1
1	2010-01-04	4	1	1	2010	2010-01-03	2010-01-01
2	2010-01-05	5	1	1	2010	2010-01-03	2010-01-01
3	2010-01-06	6	1	1	2010	2010-01-03	2010-01-01
4	2010-01-07	7	1	1	2010	2010-01-03	2010-01-01
5	2010-01-08	8	1	1	2010	2010-01-03	2010-01-01
6	2010-01-11	11	2	1	2010	2010-01-10	2010-01-01
7	2010-01-12	12	2	1	2010	2010-01-10	2010-01-01
8	2010-01-13	13	2	1	2010	2010-01-10	2010-01-01
9	2010-01-14	14	2	1	2010	2010-01-10	2010-01-01
10	2010-01-15	15	2	1	2010	2010-01-10	2010-01-01

# Ejercicio Fechas

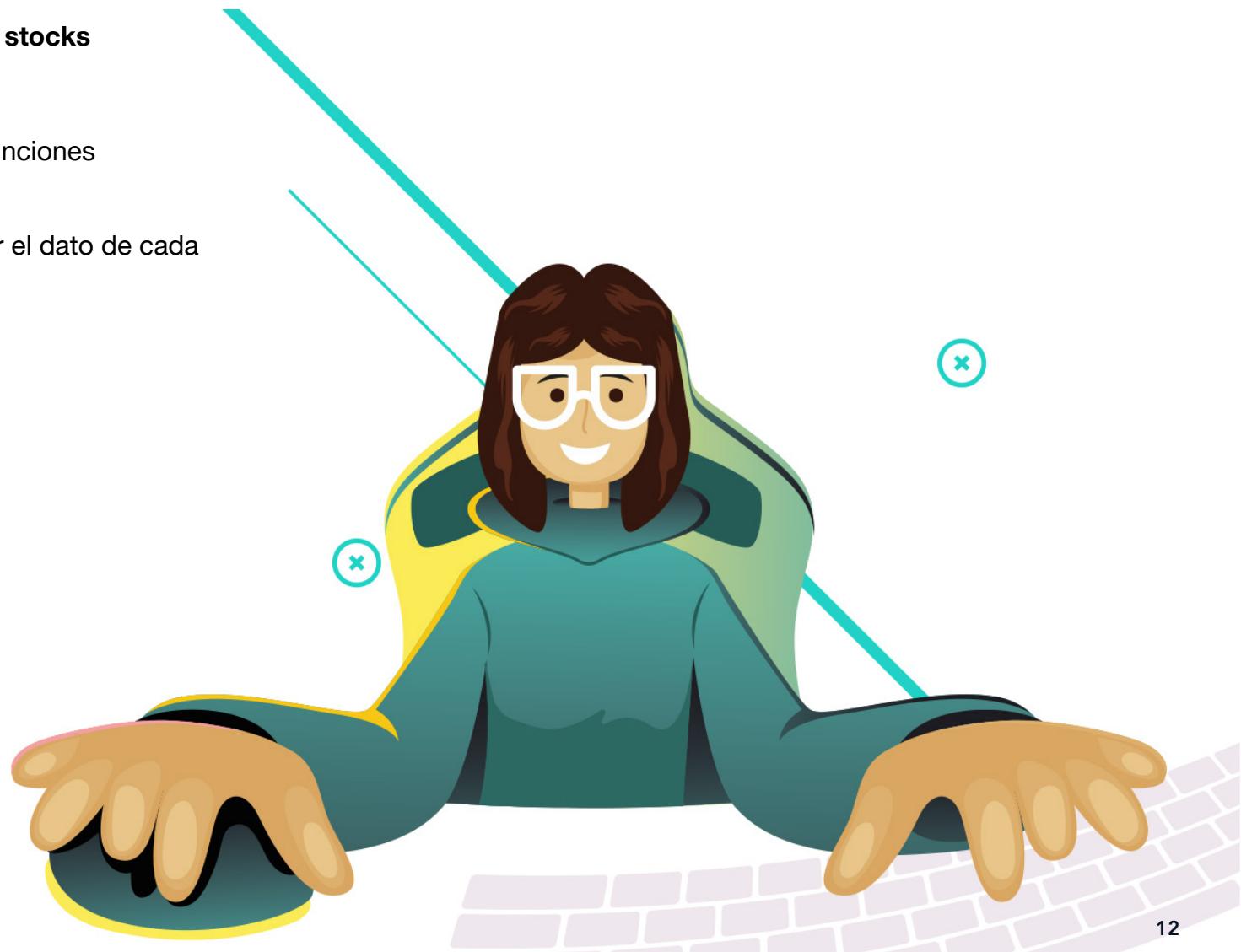
## Ejemplo sencillo

Vamos a coger la tabla de Amazon trade stocks

Comprobar si funciona en SQLite alguna de las dos funciones comentadas.(spoiler: no funciona)

Tratar de conseguir el código que nos permita obtener el dato de cada fecha desglosado.

Duración: 15 mins  
Metodología: Grupal



# Actualizar Tablas



# Actualizar Tablas

## Update

Para añadir un registro nuevo a nuestra tabla, hemos estado utilizando **INSERT INTO**, pero ¿y si queremos modificar algún registro que teníamos previamente?  
¡Podemos utilizar **UPDATE**!

```
UPDATE table_name  
SET column1 = value1,  
    column2 = value2....,  
    columnN = valueN  
WHERE [condition];
```



```
UPDATE Empleados  
SET salario = 3000.00,  
    convenio = 'CV0921'  
WHERE idemp = 'X042321';
```

## Replace

```
REPLACE(columna, 'valor_inicial', 'valor_final')
```



```
REPLACE(salario, ',', '')
```



3,000.00 pasará a 3000.00

# MongoDB



# MongoDB

¿Qué es MongoDB?

!!!!¿NO TODO ES SQL?!!!!



Es una base de Datos distribuida, gratis y de código abierto

Los documentos son almacenados en una estructura parecida a un JSON (BSON)

```
{  
    "name": "Carlos González",  
    "age": 25,  
    "city": "Alicante",  
    "cars": [  
        {  
            "model": "Renault",  
            "year": 2022,  
            "value": 20000  
        },  
        {  
            "model": "Chevrolet"  
            "year": 2009  
            "value": 10000  
        }  
    ]  
}
```

# MongoDB

## Conceptos Básicos

### Estructuras Principales de MongoDB

#### Documentos

Un registro dentro de una colección  
Es análogo a un objeto JSON (BSON)  
La unidad básica dentro de MongoDB

#### Colecciones

Agrupación de documentos  
Equivalente a una tabla en las bases de datos relacionales  
No impone un esquema

#### Base de datos

Contenedor físico de colecciones  
Cada base de datos tiene su archivo propio en el sistema de archivos  
Un cluster puede tener múltiples bases de datos

# MongoDB

## Conceptos Resumen

1. MONGO DB ES UNA **BBDD DISTRIBUIDA**, LO QUE NOS PERMITE **ESCALAR DE FORMA HORIZONTAL**.
2. EL **URI** (UNIFORM RESOURCE IDENTIFIER, “IDENTIFICADOR UNIFORME DE RECURSOS”) **IDENTIFICA UN RECURSO** POR SU NOMBRE, POR SU UBICACIÓN O POR AMBOS. EN ESTE ÚLTIMO CASO, EL URI INDICA QUE UN RECURSO IDENTIFICADO Y DÓNDE ESTÁ DISPONIBLE. COMPRENDE EL URL Y EL URN. **NO DEBE ESTAR EN NUESTRO REPOSITORIO PÚBLICO**
3. ES **BUENO ENcriptar DATOS GUARDADOS** EN NUESTRA BBDD/CLUSTER
4. LA FUNCIÓN **Pretty** MUESTRA EL RESULTADO DE UNA **QUERY** EN CONSOLA DE MANERA **MÁS AMIGABLE**
5. EN MONGODB SE PUEDEN GUARDAR **ARREGLOS DE DOCUMENTOS** Y ES UNA FORMA DE **EXPRESAR RELACIONES DE UNO A MUCHOS**.
6. SU LECTURA NO ES TAN COMPLICADA DE ENTENDER, POR EJEMPLO SI EJECUTO **DB.COLLECTION.FIND()** NOS RETORNARÁ TODOS LOS DOCUMENTOS DE LA COLECCIÓN ‘COLLECTION’
7. SI QUIERO **INSERTAR MUCHOS DOCUMENTOS** EN LA BBDD, LO PODRÍAMOS HACER CON ‘**INSERT MANY**’
8. RECUERDA QUE **MONGODB ES SCHEMA LESS** (SIN ESQUEMA)
9. EN UN CLUSTER DE MONGODB PUEDE HABER **MUCHAS BASES DE DATOS**
10. LOS DOCUMENTOS ALMACENADOS EN MONGODB NO ES NECESARIO QUE TENGAN UNA MISMA ESTRUCTURA, DE HECHO ESTA ES UNA DE SUS PRINCIPALES VENTAJAS QUE HACE QUE **MONGODB SEA MÁS ESCALABLE Y FLEXIBLE**

# Ejercicios Guiados



# Ejercicios Guiados

¡Tod@s a una!

Vamos a utilizar varias de las tablas que encontramos en *Dataset*

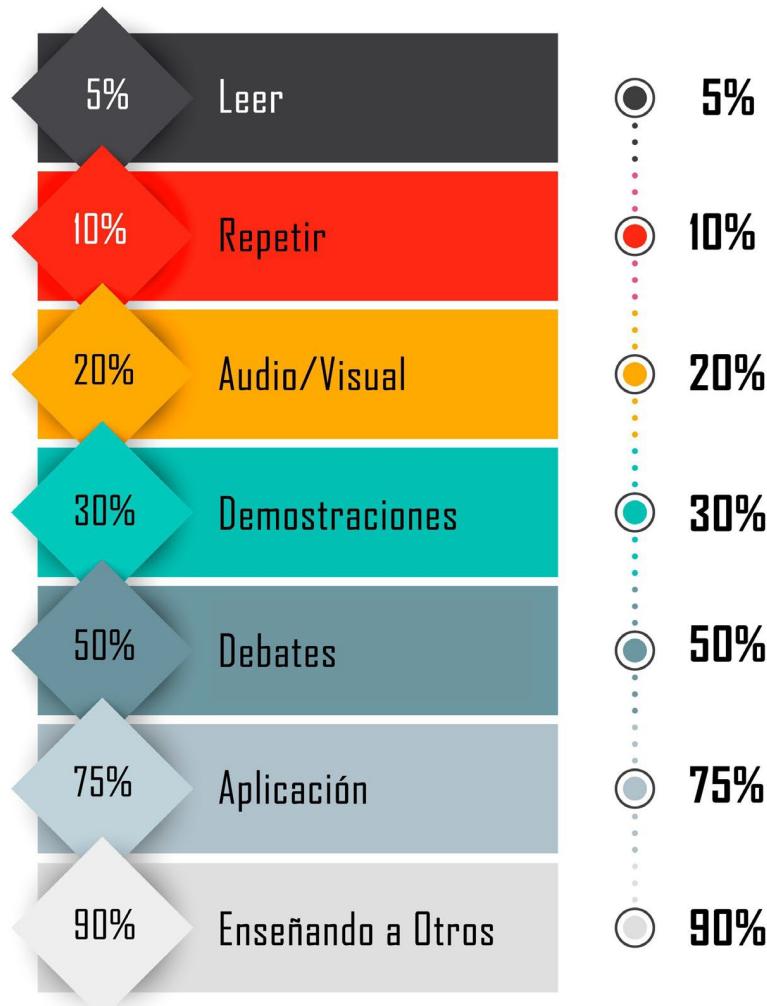
Propondremos una query para cada una de las tablas.\*

Duración: ¿? mins  
Metodología: Grupal



## ¿Por qué tanta interacción?

### La Pirámide del Aprendizaje





GOBIERNO  
DE ESPAÑA

VICEPRESIDENCIA  
PRIMERA DEL GOBIERNO

MINISTERIO  
DE ASUNTOS ECONÓMICOS  
Y TRANSFORMACIÓN DIGITAL

SECRETARÍA DE ESTADO  
DE DIGITALIZACIÓN  
E INTELIGENCIA ARTIFICIAL

red.es

Centro de  
Referencia Nacional  
en Comercio Electrónico  
y Marketing  
**CRN**  
Digital



UNIÓN EUROPEA

*“El FSE invierte en tu futuro”*

---

Fondo Social Europeo

  
**Barrabés**

 The Valley

# Fechas en SQLite

## strftime() y date()

¡El código equivalente en SQLite sería el siguiente!

**SELECT**

```
Date,
strftime('%d', Date) as day_number,
strftime('%W', Date) as week_number ,
strftime('%m', Date) as month_number ,
strftime('%Y', Date) as year_number,
date(Date, '-6 days', 'weekday 1') as week_starting_date,
date(Date, 'start of month') as month_starting_date
```

**FROM** amazon

**ORDER BY** date

## Ejercicio 2.0

Vamos a probar a utilizarlo en las tablas que tenemos de nuestras BBDD

Format	Description
%d	day of the month: 01-31
%f	fractional seconds: SS.SSS
%H	hour: 00-24
%j	day of the year: 001-366
%J	Julian day number
%m	month: 01-12
%M	minute: 00-59
%s	seconds since 1970-01-01
%S	seconds: 00-59
%w	day of week 0-6 with Sunday==0
%W	week of the year: 00-53
%Y	year: 0000-9999
%%	%