BI4SME

boosting business intelligence skills for SME growth

BI4SME - R2 —Materiales de Capacitación Unidad 2: SQL suficiente paraBI

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

Contents

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

UNIDAD 2: SQL suficiente para BI

2.1. Introducción y configuración del entorno

Alcance de la capacitación

Bienvenido a la capacitación básica de SQL. Este curso cubrirá los fundamentos del Lenguaje de Consulta Estructurada (SQL), un poderoso lenguaje utilizado para acceder y gestionar datos contenidos en bases de datos. A lo largo de este curso, cubriremos la sintaxis básica y los comandos utilizados para consultar bases de datos y algunas de las características más avanzadas de SQL, como subconsultas y joins. Al final de este curso, entenderás completamente los fundamentos del análisis consultando bases de datos.

0. **Lenguaje de Definición de Datos (DDL):** Utilizado para crear, modificar y eliminar objetos de bases de datos. Ejemplos: CREATE, ALTER, DROP.
1. **Lenguaje de Manipulación de Datos (DML):** Utilizado para manipular datos en una base de datos. Ejemplos: SELECT, INSERT, UPDATE, DELETE.
2. **Lenguaje de Control de Datos (DCL):** Utilizado para controlar el acceso de usuarios a una base de datos. Ejemplos: GRANT, REVOKE.
3. **Lenguaje de Control de Transacciones (TCL):** Utilizado para gestionar transacciones en una base de datos. Ejemplos: COMMIT, ROLLBACK.

Como se mencionó anteriormente, este curso trata sobre los comandos utilizados para realizar consultas en las bases de datos contenidas en la categoría DML.

Este no es un curso sobre modelado de bases de datos, la creación y administración de una base de datos son temas que están fuera del alcance de este curso y requerirán una capacitación especializada. Asumiremos que

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

estás o estarás trabajando en una base de datos existente y tu trabajo es analizar los datos almacenados en ella.

Qué es una base de datos SQL?

Una base de datos relacional almacena datos en tablas que contienen filas y columnas, también conocidas como relaciones. Estas tablas están relacionadas entre sí mediante el uso de columnas comunes que se utilizan para enlazar las tablas. Un sistema de gestión de bases de datos relacional (RDBMS) permite a los desarrolladores y administradores crear, actualizar y gestionar los datos almacenados en la base de datos. SQL es el lenguaje utilizado para consultar y gestionar datos en una base de datos relacional.

En este curso utilizaremos dos bases de datos simples: **"db_books"** y **"join_tables".** Aprenderás cómo instalar un motor de base de datos y cómo crear estas dos bases de datos.

2.1.1. Instalación de MariaDB

Para seguir este curso, necesitas tener un gestor de base de datos SQL relacional instalado en tu computadora. De las muchas opciones disponibles en el mercado, utilizamos MariaDB en nuestro curso de capacitación. MariaDB tiene varias ventajas:

- El servidor MariaDB es una de las bases de datos relacionales de **código abierto más populares**. Está hecho por los desarrolladores originales de MySQL y está garantizado que permanecerá como código abierto. Forma parte de la mayoría de las ofertas en la nube y viene por defecto en la mayoría de las distribuciones de Linux.
- Es totalmente gratuito.
- Podemos operar en la base de datos MariaDB a través de Heidi, una interfaz de usuario fácil de usar e intuitiva.

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

Enlaces de interés

- Descarga de MariaDB: https://mariadb.com/downloads/
- Guía de instalación de MariaDB en Windows: https://mariadb.com/kb/en/installing-mariadb-msi-packages-on-windows/
- Tutorial básico de MariaDB: https://mariadb.com/kb/en/mariadb-basics/
- Script de creación de tablas SQL: db_books.sql

Instalación de MariaDB

MariaDB tiene un asistente de instalación fácil. Para instalar este motor de base de datos no diferirá del proceso de instalación de otros programas en Windows. Una vez finalizado el proceso, el motor de MariaDB y Heidi SQL, una interfaz de usuario SQL para interactuar con MariaDB, estarán listos en tu computadora.

1. Busca MariaDB

Busca en Google 'install MariaDB' o simplemente ve directamente al siguiente enlace:
https://mariadb.com/downloads/

2. Descarga el Instalador

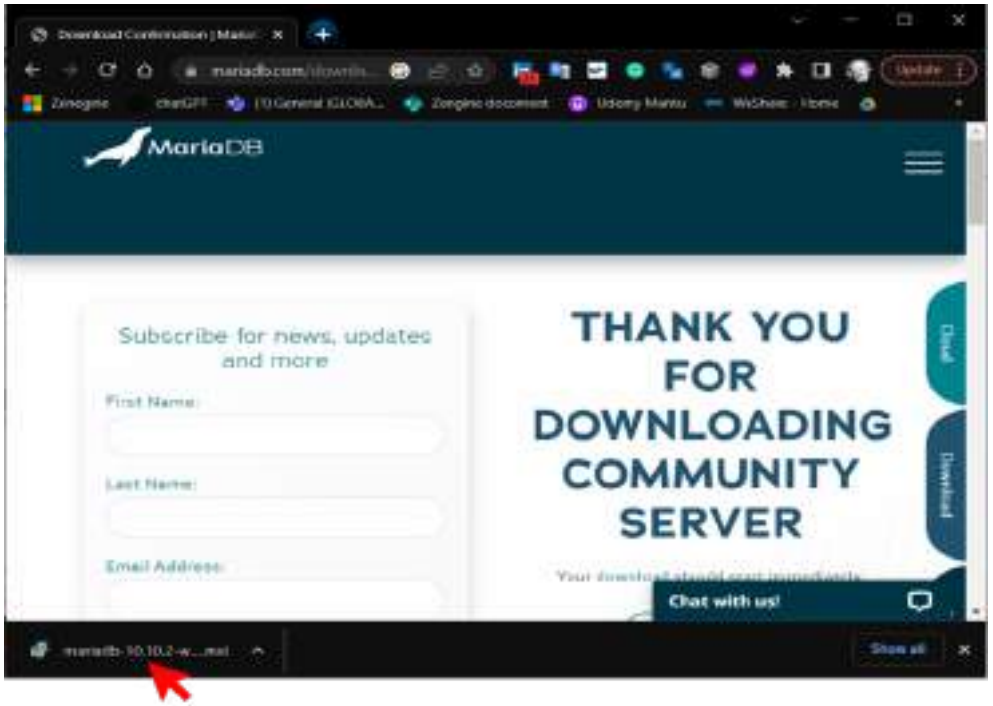
Selecciona la pestaña "Maria DB Community" tab

Asumiendo que estás usando una versión de Windows de 64 bits. En la lista desplegable selecciona MS Windows (64-bit).

Una vez seleccionado MS Windows, haz clic en Descargar.

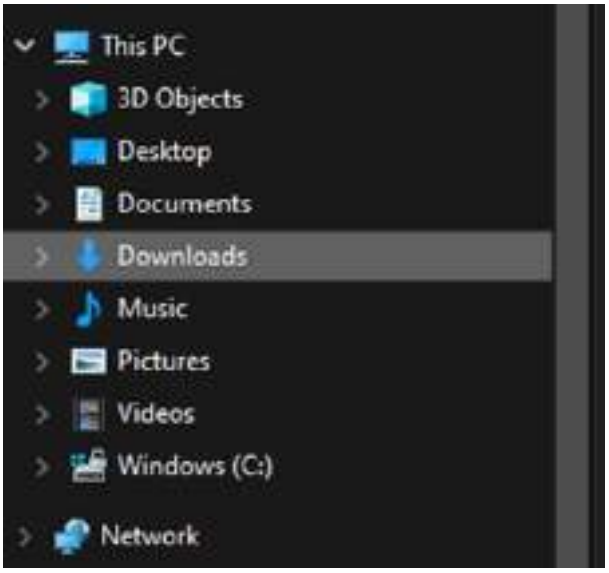
3. Ejecuta el programa instalador

Encontrarás tu instalador en la barra de descargas de tu navegador o en el explorador de archivos de tu computadora.



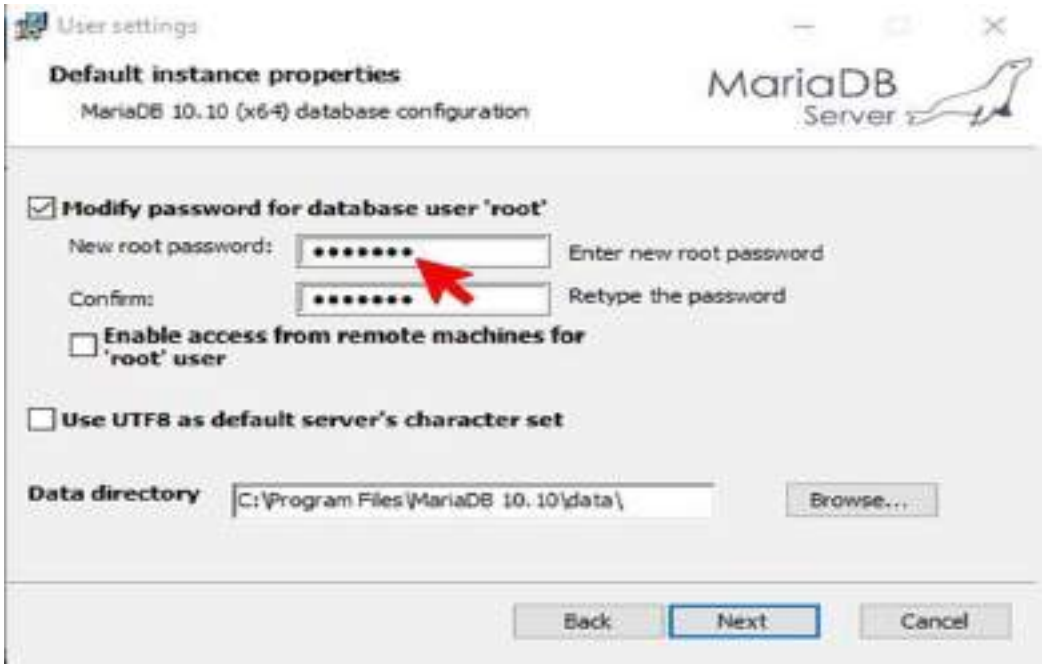
O:

Co-funded by the European Union
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132



Simplemente haz doble clic en el programa descargas y comenzará el proceso de instalación.
Para una instalación clásica, simplemente confirma todos los pasos haciendo clic en Siguiente.

En esta pantalla se te pedirá que introduzcas una contraseña para el usuario root, el usuario administrador de la base de datos. **Asegúrate de no olvidar tu contraseña.**

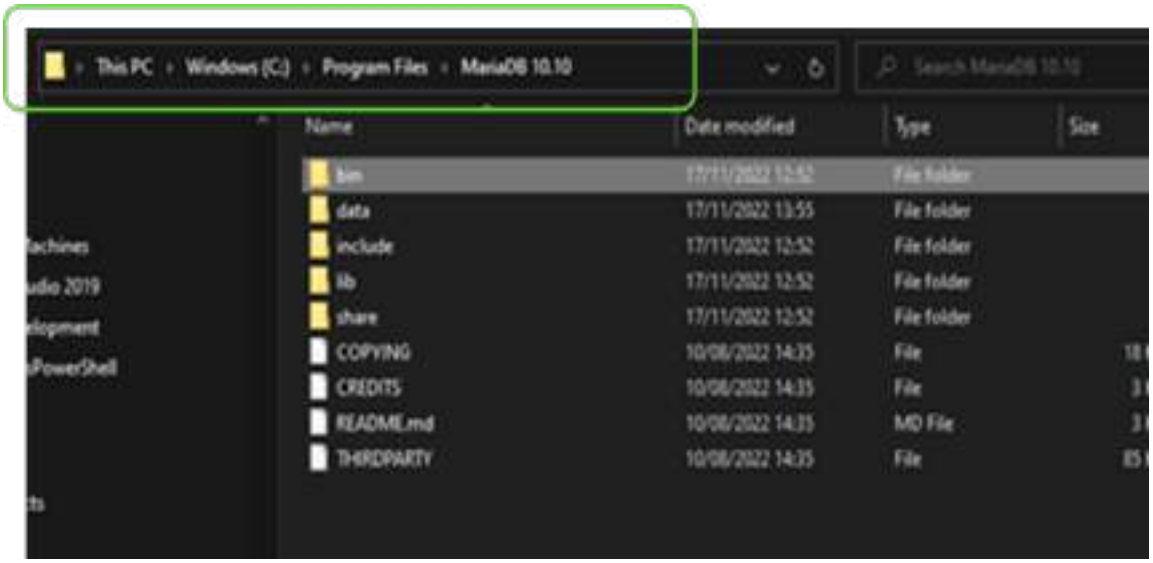

User settings
Default instance properties
MariaDB 10.10 (x64) database configuration
MariaDB Server
Modify password for database user 'root'
New root password:
Enter new root password
Confirm:
Retype the password
Enable access from remote machines for 'root' user
Use UTF8 as default server's character set
Data directory
C:\Program Files\MariaDB 10.10\data\
Browse...
Back
Next
Cancel

Continúa haciendo clic en “Siguiente” para los pasos posteriores.
Listo para instalar? Haz clic en "Instalar".

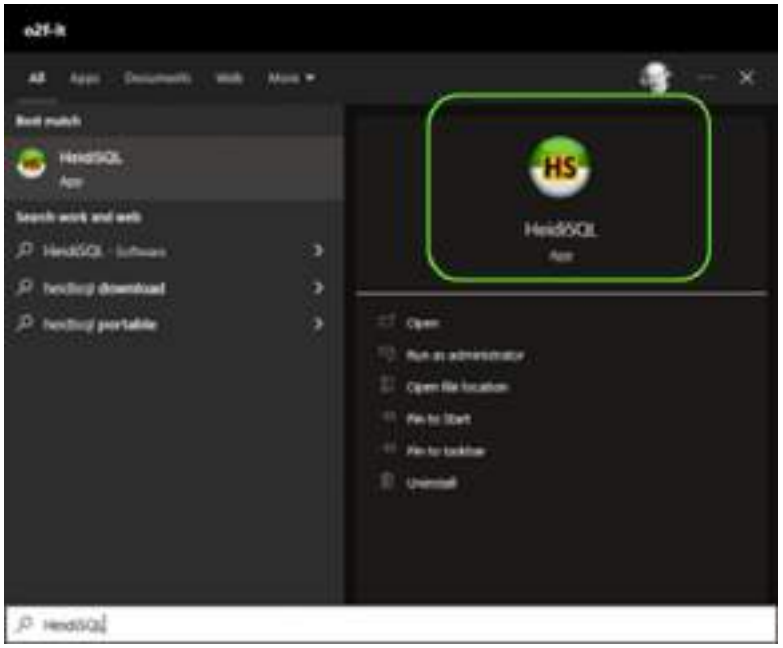
Fin

4. Verifica tu instalación

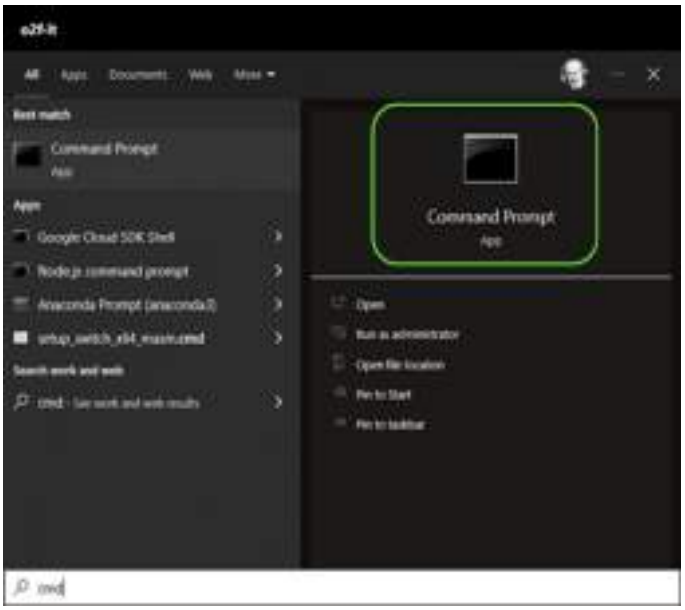
Si no has cambiado la ruta de instalación predeterminada, verás una nueva carpeta en tu PC en C:/Program Files/MariaDB <versión>



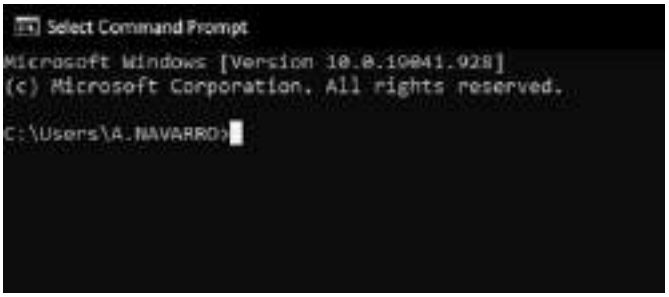
También estará disponible un nuevo programa: "HeidiSQL"



Vamos a iniciar MariaDB para comprobar si está funcionando correctamente.
Primero necesitamos usar el símbolo del sistema de Windows. No te preocupes, solo serán unas pocas líneas, ¡prometido!
Para abrir el "Terminal" o símbolo del sistema, solo escribe cmd en la barra de búsqueda de Windows.



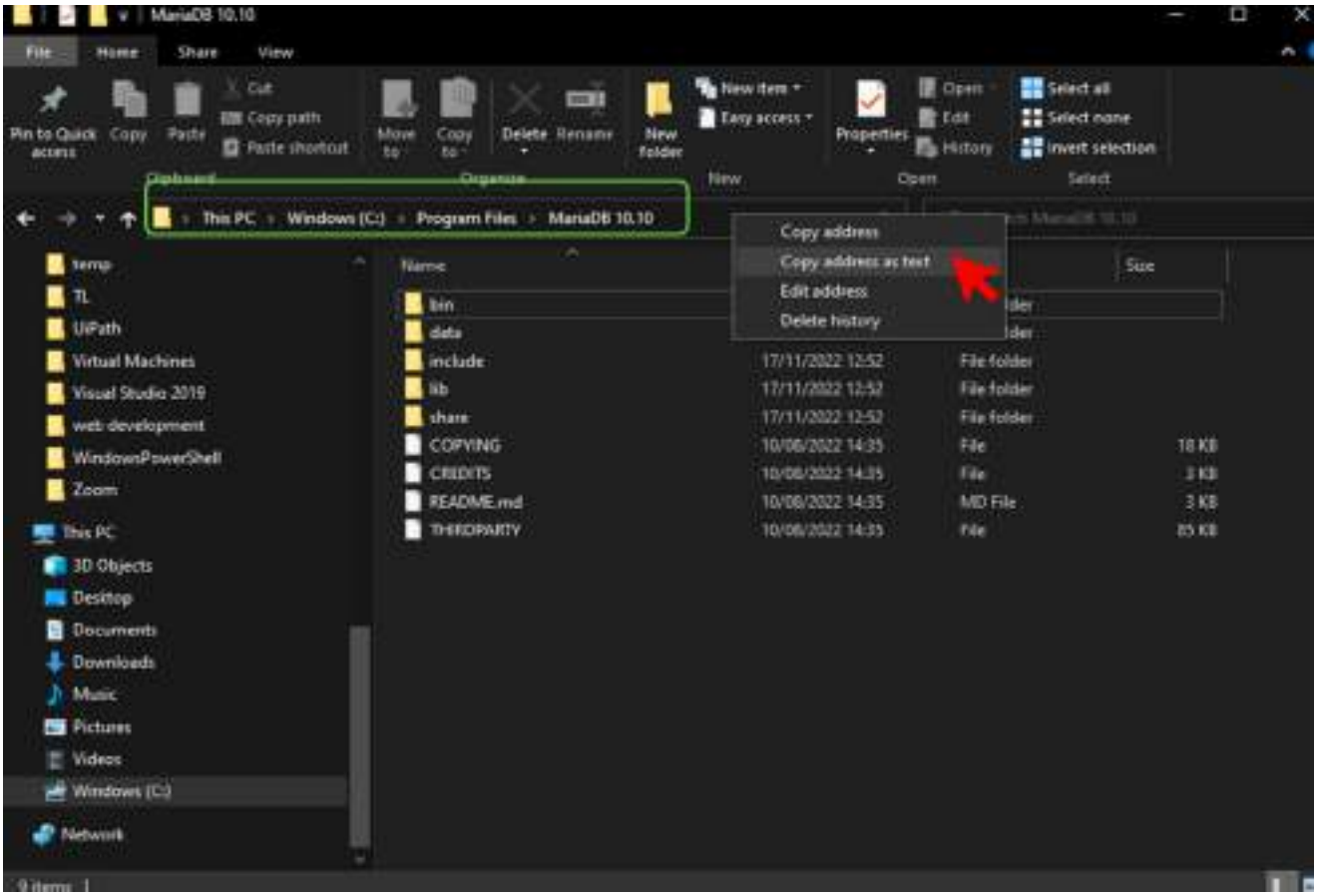
Una vez abierto el Terminal, debes moverte a la carpeta donde se instaló MariaDB.



NOTE: Nota: Mi ruta por defecto en el símbolo del Sistema es C:\Users\A.NAVARRO, pero necesito moverme a la ruta C:/Program Files/MariaDB 10.10.

Co-funded by the European Union
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

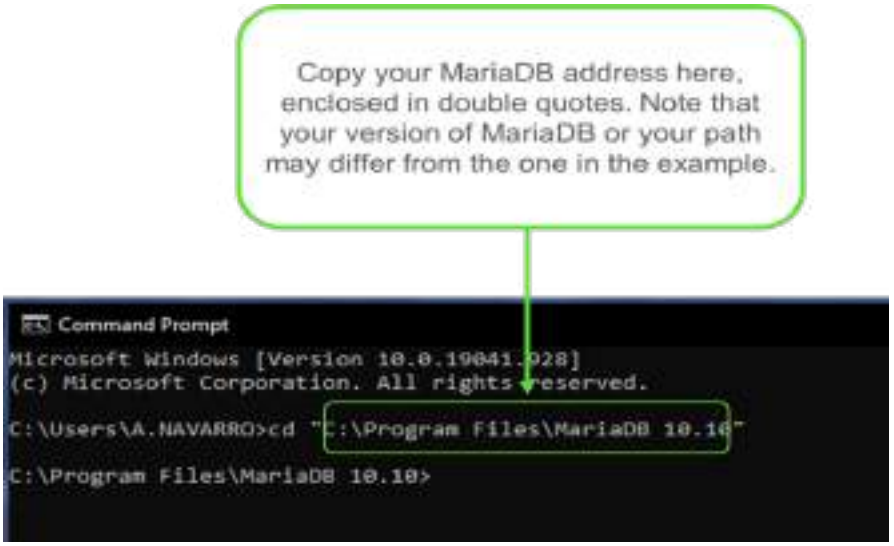
1. *Ve a mi explorador de archivos y copia la ruta de instalación de MariaDB.*



2. *Ahora escribe el siguiente comando en tu terminal cd "<paste your MariaDB route here>"*

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

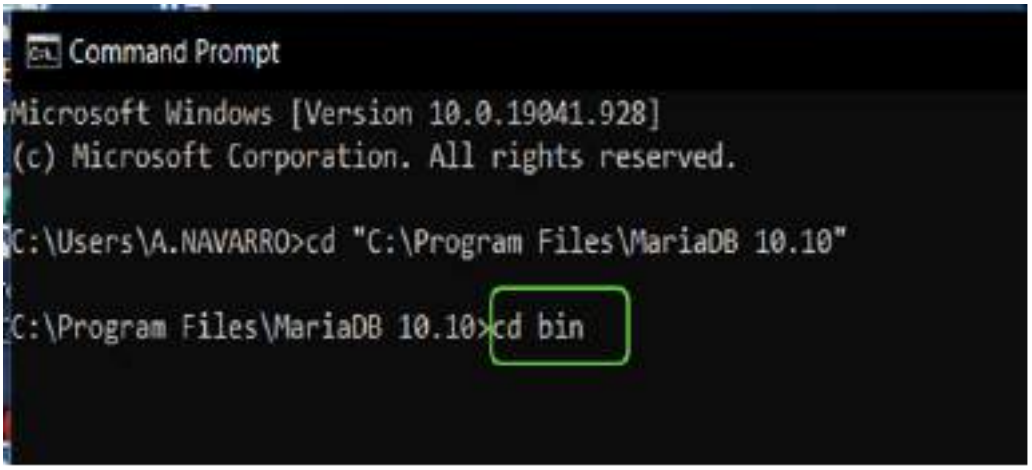


Click Enter.

3. *Para iniciar el motor de base de datos, ve a la carpeta bin:*

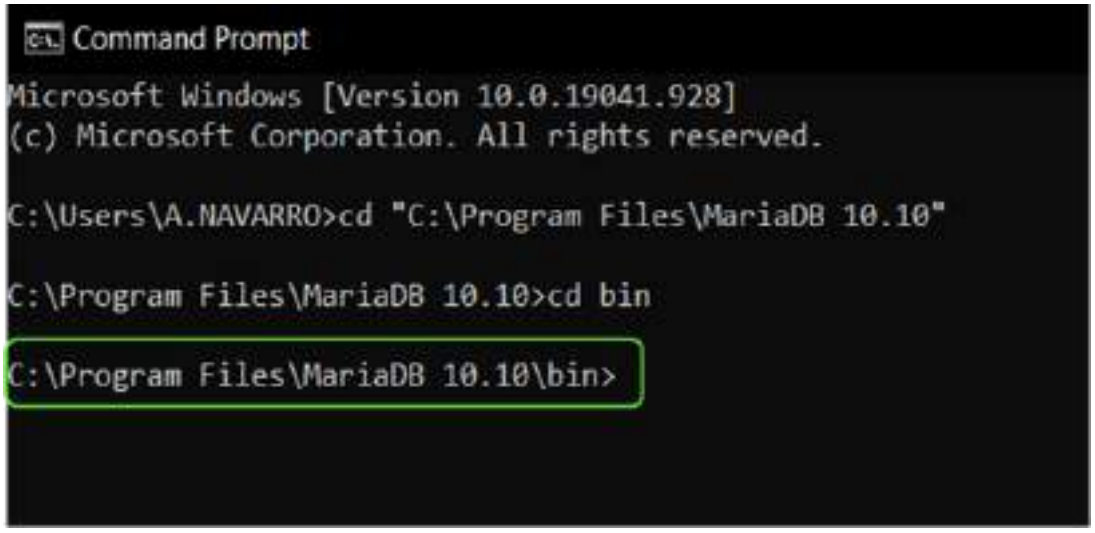
Escribe:

cd bin



Presiona enter.

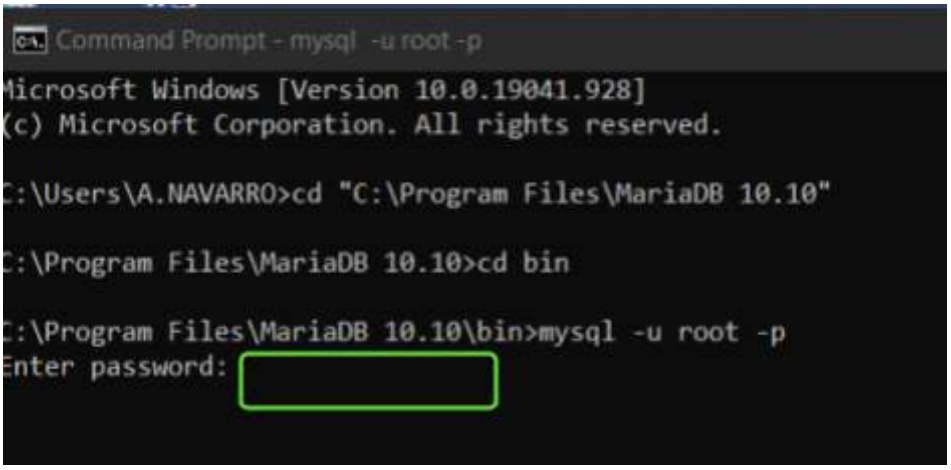
Co-funded by the European Union
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132



Escribe:

> mysql -u root -p

y presiona Enter. MariaDB te pedirá la contraseña que usaste durante la fase de instalación.



Introduce tu contraseña y presiona **ENTER**

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

Muestra DATABASES

Verifica las bases de datos en nuestro motor DB.
Escribe: **show databases;**
MariaDB [(none)]> **show databases;**

When we use MariaDB directly on the command prompt it is important to end all the commands with ";" before clicking ENTER..

Verás las bases de datos creadas por defecto en nuestro motor DB.

Crear una nueva base de datos

Debemos crear una nueva base de datos con el nombre “newDB”

Co-funded by the European Union
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

MariaDB [(none)]> **CREATE DATABASE newDB;**

Note that SQL is not case sensitive, you can write sentences in uppercase or lowercase letters.

MariaDB [(none)]> **CREATE DATABASE newDB1;**

```
MariaDB [(none)]> CREATE DATABASE newDB;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> create database newDB1;
Query OK, 1 row affected (0.001 sec)
```

Ahora verás las nuevas bases de datos creadas.

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| newdb              |
| newdb1             |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.003 sec)

MariaDB [(none)]>
```

DROP DATABASE

Como no utilizaremos newDB y newDB1, es mejor eliminarlas para mantener nuestro entorno de trabajo limpio.

MariaDB [(none)]> **DROP DATABASE newdb**

```
MariaDB [(none)]> DROP DATABASE newdb;
Query OK, 0 rows affected (0.010 sec)

MariaDB [(none)]> DROP DATABASE newdb1;
Query OK, 0 rows affected (0.011 sec)

MariaDB [(none)]>
```

Crear la base de datos para seguir el curso de SQL

Utilizaremos la base de datos db_books para seguir las diferentes lecciones de nuestro curso de SQL.

Creamos el database "db_books"

MariaDB [(none)]> **CREATE DATABASE db_books;**

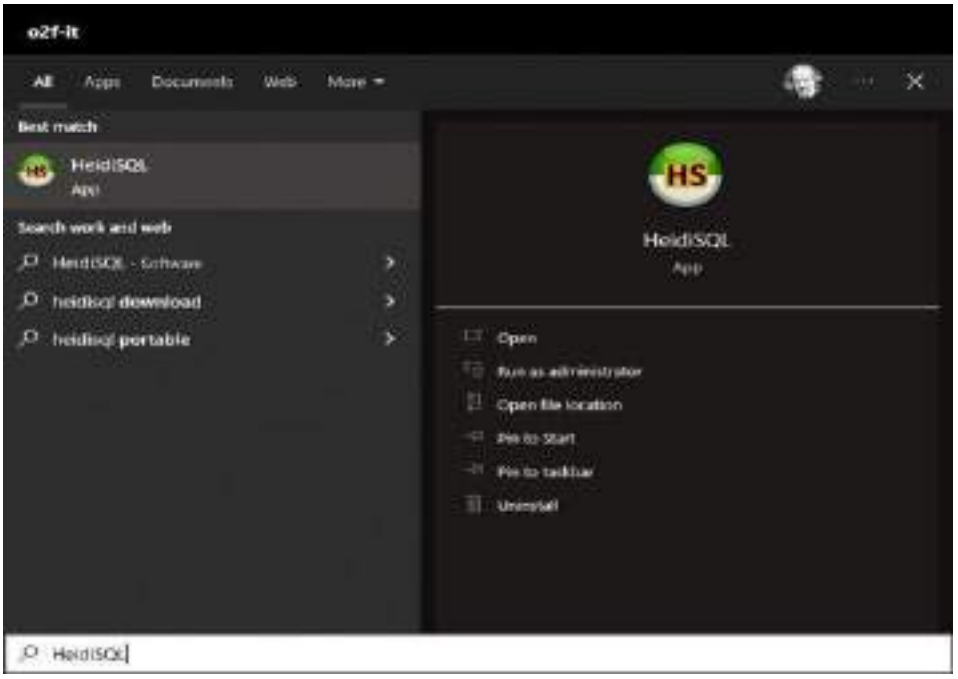
Salir de MariaDB

Para salir de MariaDB en el terminal, simplemente haz clic en: Ctrl+C

2.1.2. HeidiSQL User interfaz

Trabajar con el símbolo del sistema con MariaDB es perfectamente válido ya que todas las sentencias y consultas SQL pueden ejecutarse directamente en el terminal de Windows. Sin embargo, esto no es fácil de usar y el proceso de depuración puede ser tedioso. Existen entornos de desarrollo integrados (IDEs) que ayudan a los desarrolladores a crear su código. En nuestro curso de SQL, utilizaremos HeidiSQL, un IDE de código abierto fácil de usar y con muchas características que nos ayudarán a crear nuestras sentencias SQL

HeidiSQL se instaló durante el proceso de instalación de MariaDB. Encuentra Heidi buscando en la barra de búsqueda de Windows.



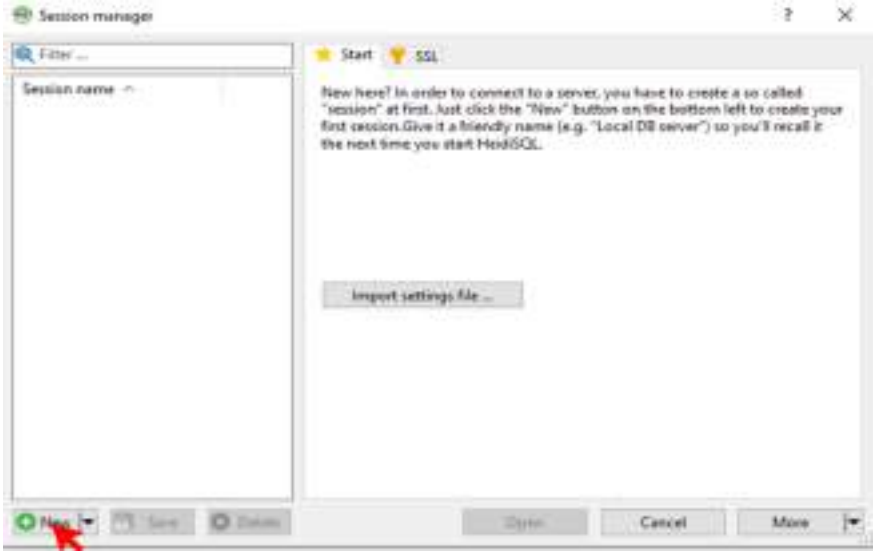
Haz clic en el icono y abre el programa.

Enlaces de interés

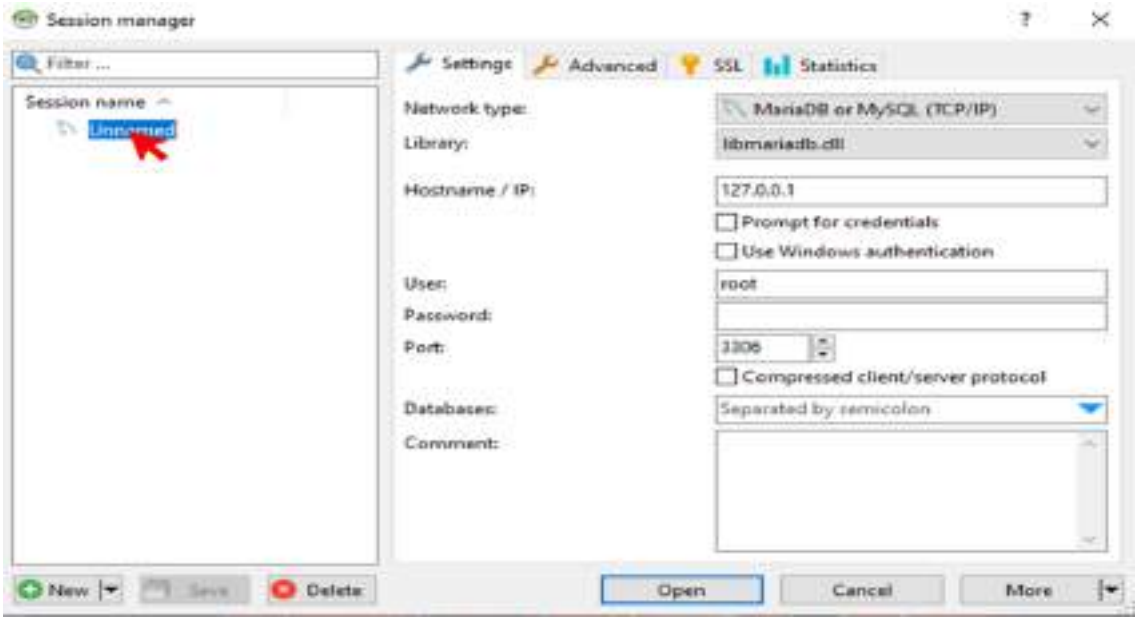
Ayuda básica sobre cómo usar
HeidiSQL: https://www.heidisql.com/help.php

Abrir HeidiSQL por primera vez

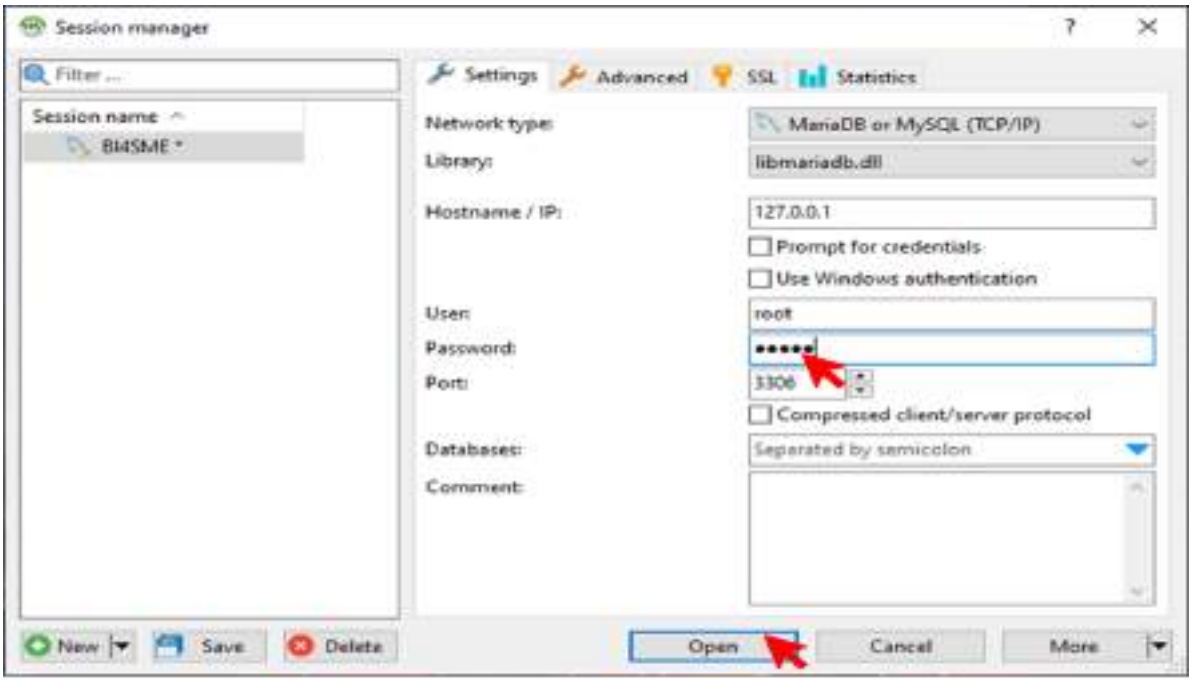
La primera vez que abrimos Heidi, se debe crear una nueva “Sesión”



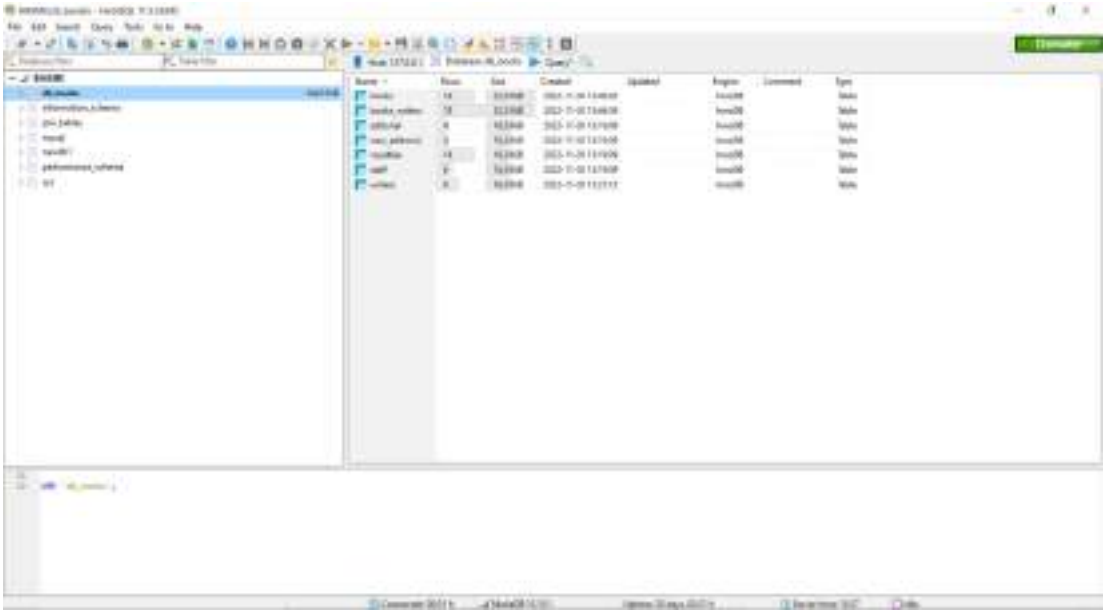
Verás que la nueva sesión tiene por defecto el nombre "Unnamed". El texto está resaltado en azul, haz clic en él y renómbralo, por ejemplo, a "BI4SME".



Después de renombrar la sesión, simplemente introduce la contraseña. La contraseña es la misma que la utilizada durante el proceso de instalación de MariaDB.

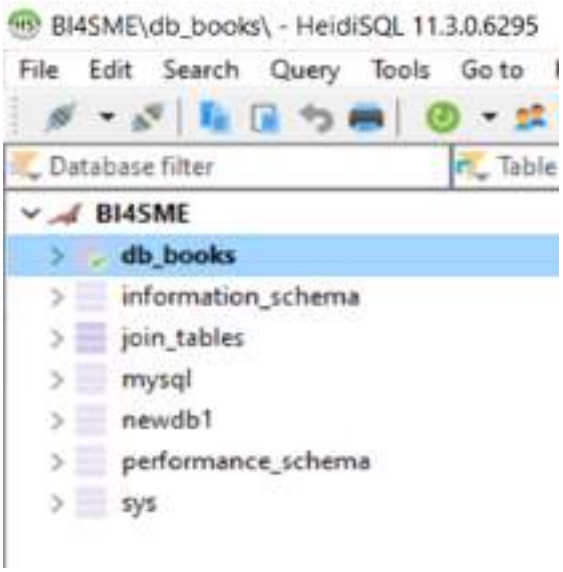


Al hacer clic en “Open” llegarás a esta pantalla:



Información General sobre la Interfaz

Veamos cómo las bases de datos que vimos anteriormente en el Command Prompt están listadas justo debajo de la sesión abierta.



Podemos confirmar que la base de datos db_books está presente con un tamaño de 0B, lo que tiene sentido ya que esa base de datos está

Co-funded by the European Union
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

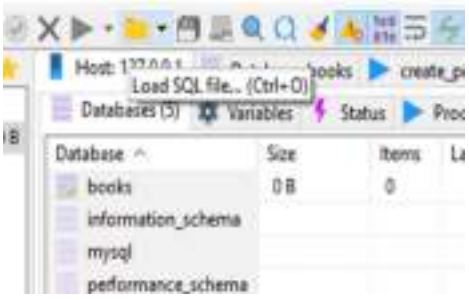
completamente vacía. No hemos creado ninguna tabla ni ingresado datos en la base de datos aún.

Puedes tener una visión general de las herramientas de Heidi aquí (https://www.heidisql.com/help.php); nos limitaremos en esta capacitación a funciones muy básicas de este IDE.

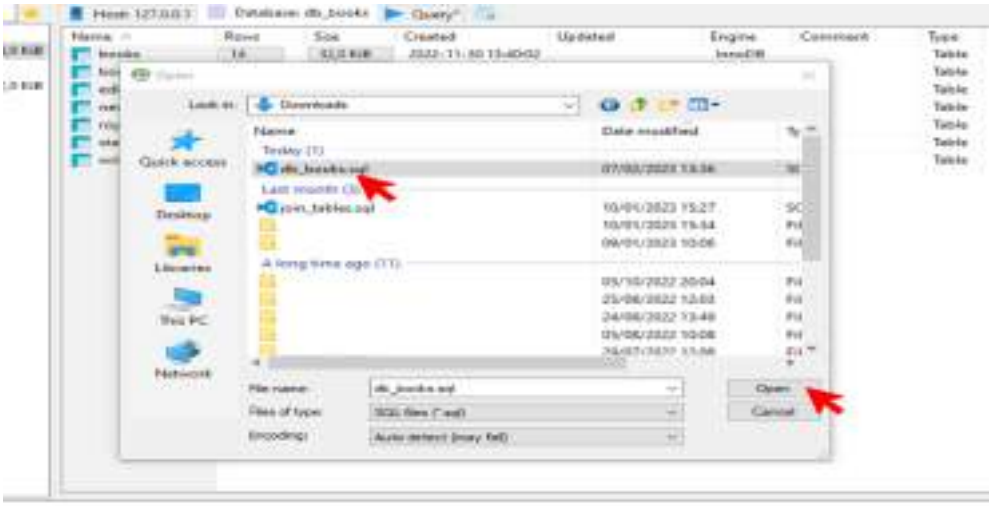
Vamos a agregar contenido a nuestra base de datos de libros. Afortunadamente, tenemos el script SQL db_books.sql adjunto en esta lección con todas las sentencias necesarias para crear las tablas, agregar las relaciones entre ellas y registrar los datos en las diferentes tablas. Por lo tanto, lo único que debemos hacer es ejecutar el script con Heidi SQL

- Descarga el archivo db_books.sql

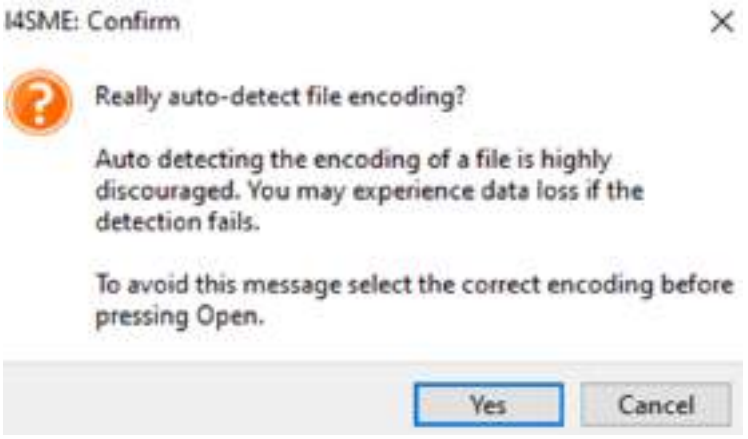
- En HeidiSQL, ve al icono de cargar archivo en la barra de herramientas superior o simplemente presiona Ctrl + O.



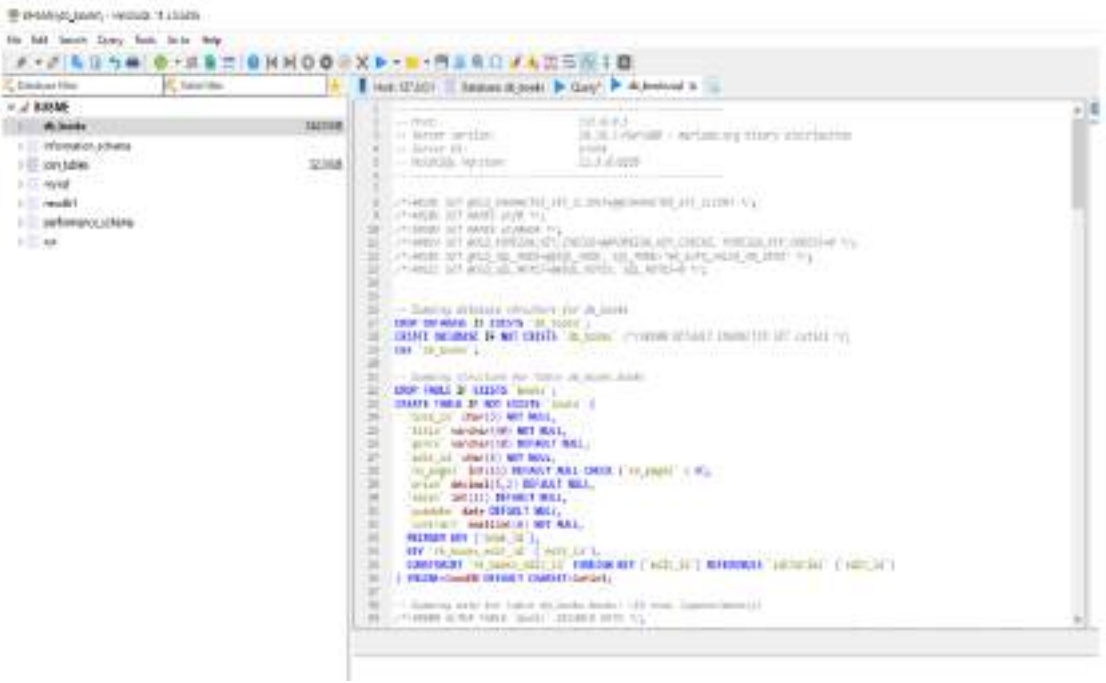
En el popup, navega a la carpeta donde se descargó el archivo db_books.sql. En este caso, el archivo se guardó en la carpeta de descargas. Simplemente selecciona el archivo haciendo clic en el icono del archivo.



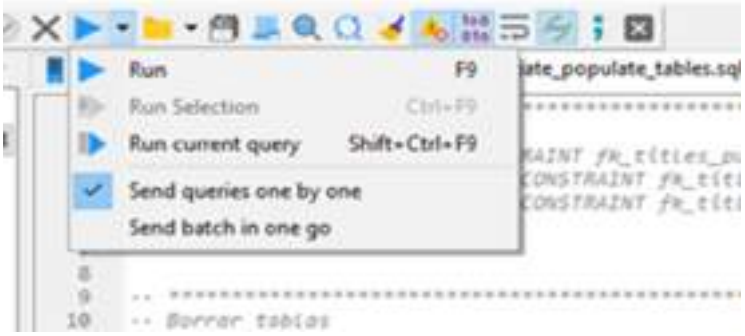
- Haz clic en "Open" y descarta esta advertencia.



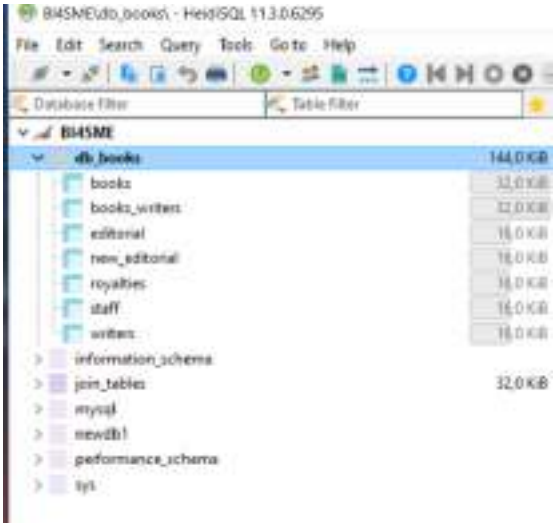
- Observa cómo las líneas del script ahora están cargadas en el editor de consultas.



- Para ejecutar la consulta, simplemente haz clic en RUN o presiona F9.

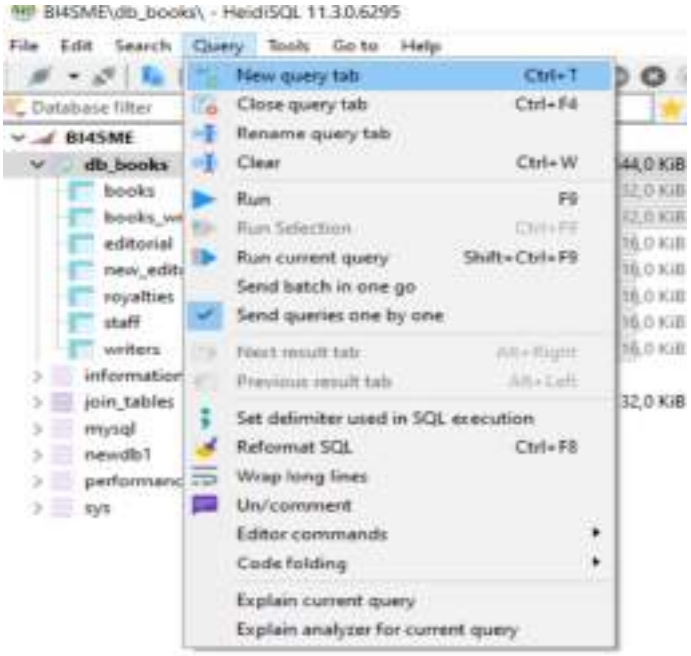


Ahora puedes ver las nuevas tablas creadas en db_books.



Nuestro primero QUERY

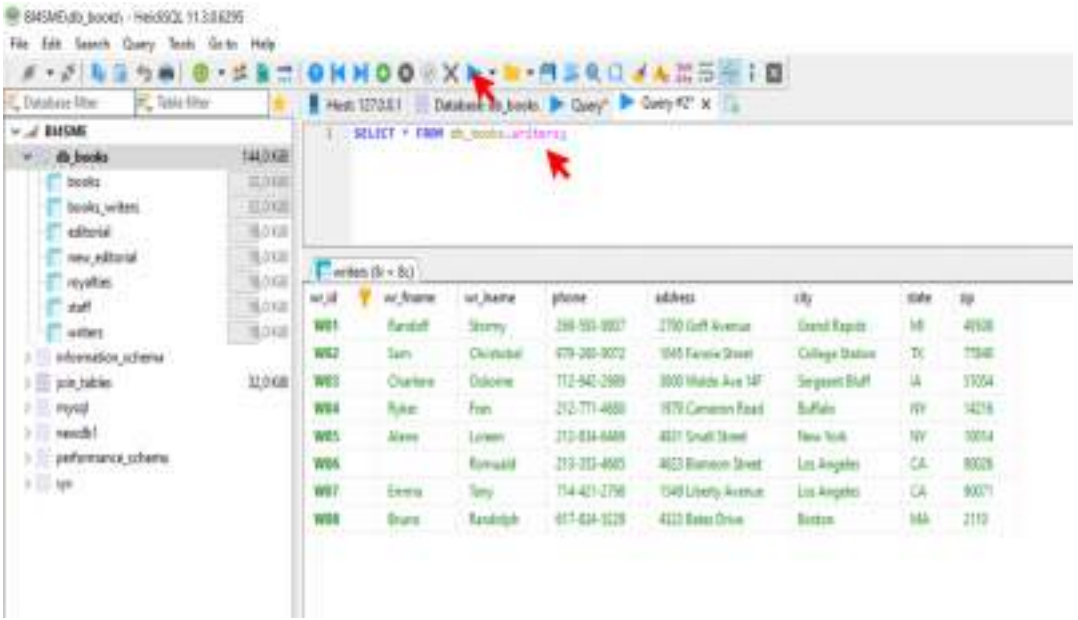
Ve a Query en la pestaña del menú superior y selecciona “New query tab” o simplemente escribe Ctrl + T.



O escribe CYRL + T

En la nueva pestaña de consulta, simplemente escribe:

```
SELECT * FROM db_books.writers;
```



Felicidades! ¡Has realizado tu primer Query!

2.2.1. The SELECT keyword

"En las siguientes unidades, aprenderás la sintaxis básica de las consultas y comandos más avanzados para mejorar el análisis de los datos incluidos en una tabla de base de datos. Por supuesto, SQL nos permite consultar y relacionar datos de más de una tabla, pero eso se tratará en el próximo bloque de unidades.

Al final de esta sección, podrás:

- Seleccionar y filtrar datos de una tabla.
- Realizar consultas de agrupación.
- Filtrar datos duplicados.
- Trabajar con datos nulos.

Como de costumbre, explicaremos los conceptos de manera práctica a través de ejemplos y ejercicios.

¡Prepárate para desbloquear el poder de SQL!" 😊

- **SELECT**: Define el campo, las columnas que se incluirán en la consulta.

Normalmente, un SELECT siempre va acompañado de una declaración FROM.

- **FROM:** Especifica la tabla de origen.
  La palabra clave FROM no es obligatoria si no nos referimos a ninguna tabla de la base de datos.

Ve el ejemplo a continuación de un script SQL básico; como las columnas en los datos seleccionados no se refieren a ninguna tabla en la base de datos, la palabra clave FROM no está presente.

```
SELECT "Madrid" AS 'location'
       , current_date() AS 'date';
```

```
+----------+------------+
| location | date       |
+----------+------------+
| Madrid   | 2023-01-23 |
+----------+------------+
```

SELECT columns FROM TABLES

Sintaxis Básica

Para consultas a la base de datos, la declaración más simple incluirá las palabras clave **SELECT y FROM.**

Después de la palabra clave **SELECT** indicaremos los nombres de las columnas, los nombres de las columnas se separan por una coma.

Después de la palabra clave **FROM** añadiremos la tabla que contiene las columnas definidas en el **SELECT**.

Por ahora estamos escribiendo consultas que solo involucran una sola tabla:

```
SELECT column_1, column_2, …, column_n
FROM table_name
```

Nuestra primera query simple**: *"LIST THE BOOK NAMES AND ITS PRICE":***

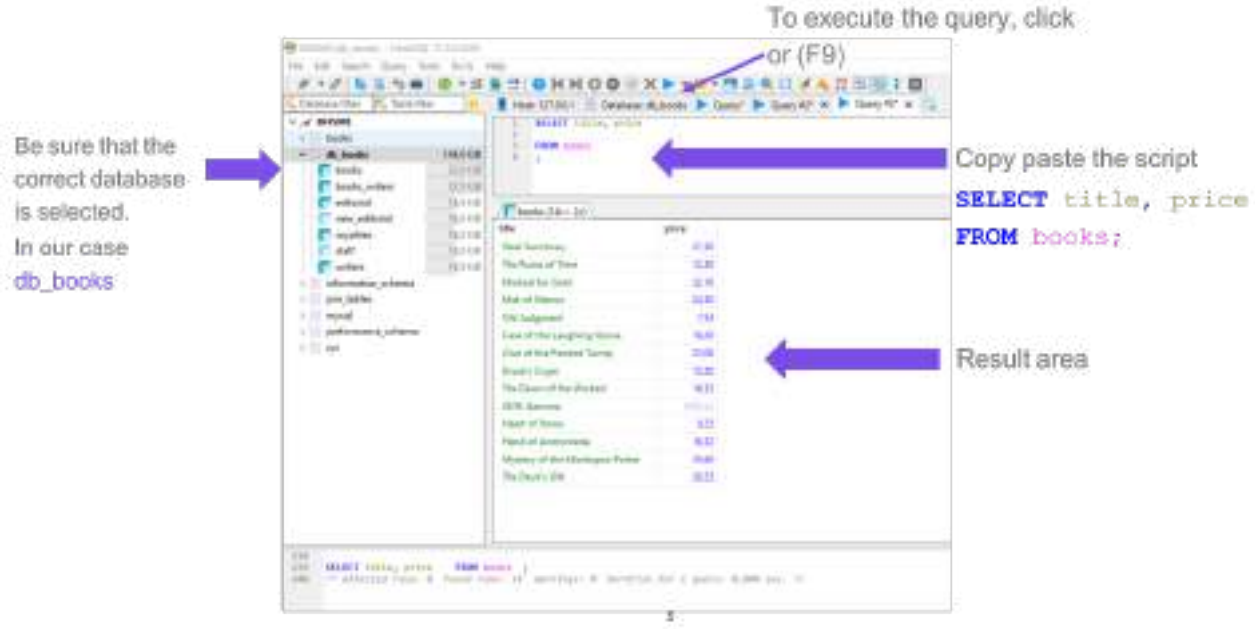
Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

```
SELECT title, price

FROM books
```

```
+----------------------------------+-------+
| title                            | price |
+----------------------------------+-------+
| Steel Sanctuary                  | 21.30 |
| The Ruins of Time                | 12.30 |
| Marked for Gold                  | 32.10 |
| Mak of Silence                   | 24.30 |
| Old Judgment                     |  7.53 |
| Case of the Laughing Goose       | 18.20 |
| Clue of the Painted Turnip       | 25.00 |
| Druid's Crypt                    | 13.20 |
| The Dawn of the Wicked           | 16.32 |
| 2619: Gamma                      |  NULL |
| Heart of Stone                   |  9.25 |
| Hand of Andromeda                | 16.32 |
| Mystery of the Misshapen Porter  | 35.60 |
| The Devil's Gift                 | 29.23 |
+----------------------------------+-------+
```

Simplemente copia y pega este script en el editor de consultas de MariaDB.



El "DOT" NOTATION

Escribimos la consulta anterior con una sintaxis simplificada:

```
SELECT column_1, columna_2
FROM table_name;
```

```
SELECT title, price
FROM books;
```

In fact, the correct way to write queries should follow this pattern:

```
SELECT
database.table_name.column_1,
database.table_name.columna_2

FROM database.table_name;
```

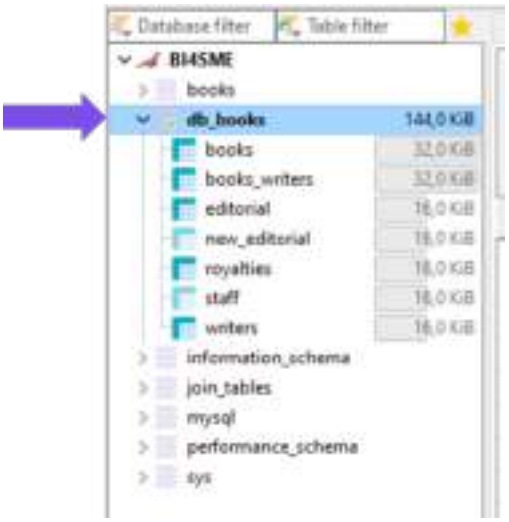
```
SELECT
db_books.books.title,
db_books.books.price

FROM db_books.books;
```

De hecho, la forma correcta de escribir consultas debería seguir este patrón:

database_name.table_name.column_name = column_path
database_name.table_name=table_path

La razón por la cual la notación anterior funciona es porque seleccionamos en la interfaz de usuario de Heidi la base de datos, por lo que SQL sabe dónde buscar, las columnas y las tablas.

Para simplificar, siempre utilizaremos la notación corta en este curso, pero recuerda siempre que la base de datos correcta debe estar seleccionada en tu editor de Heidi.



Si estás utilizando MariaDB directamente en un terminal en tu PC, recuerda seleccionar la base de datos con el comando USE

```
MariaDB [(none)]> use db_books
Database changed
MariaDB [db_books]>
```

SELECT – CASO DE TABLAS Y COLUMNAS

SELECT * : Selecciona todos los campos en la tabla.

SELECT column_1, column_2, ...: Indica nombres de columnas separados por comas.

En SQL, los espacios o saltos de línea no afectan la consulta. Por lo tanto, la siguiente notación también es válida:

```
SELECT column_1 ,
       column_2 ,
...
```

Nuestra notación recomendada "comas al inicio" (facilita eliminar una columna simplemente eliminando una línea; el resto de la consulta puede ejecutarse sin modificaciones adicionales):

```
SELECT column_1
     , column_2
     ,...
```


Co-funded by the European Union
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

FROM – CASO de TABLES

Se refiere a la tabla donde estamos ejecutando la consulta. Usualmente, la tabla se define por la notación:
FROM database_name.table_name

Por ejemplo, si queremos seleccionar la tabla "titles" de nuestra base de datos "books", la notación debería ser:
FROM books.titles

Como ya se mencionó, la notación con 'punto' puede simplificarse si seleccionamos la base de datos previamente en el terminal de MariaDB o en la interfaz de usuario de Heidi.
En una notación simplificada, la sintaxis es:
: **FROM** titles

CÓMO RESOLVER UN QUERY SQL, CÓMO PENSAR

Supongamos que se nos pide obtener el siguiente informe:

"Obtener una tabla con el título del libro, su tipo y el número de páginas que contiene."

Para obtener la información solicitada, debemos hacernos las siguientes preguntas:

- En qué base de datos tenemos la información que estamos buscando?
- Qué campos contienen la información?
- Qué responde a la siguiente pregunta, en qué tabla?

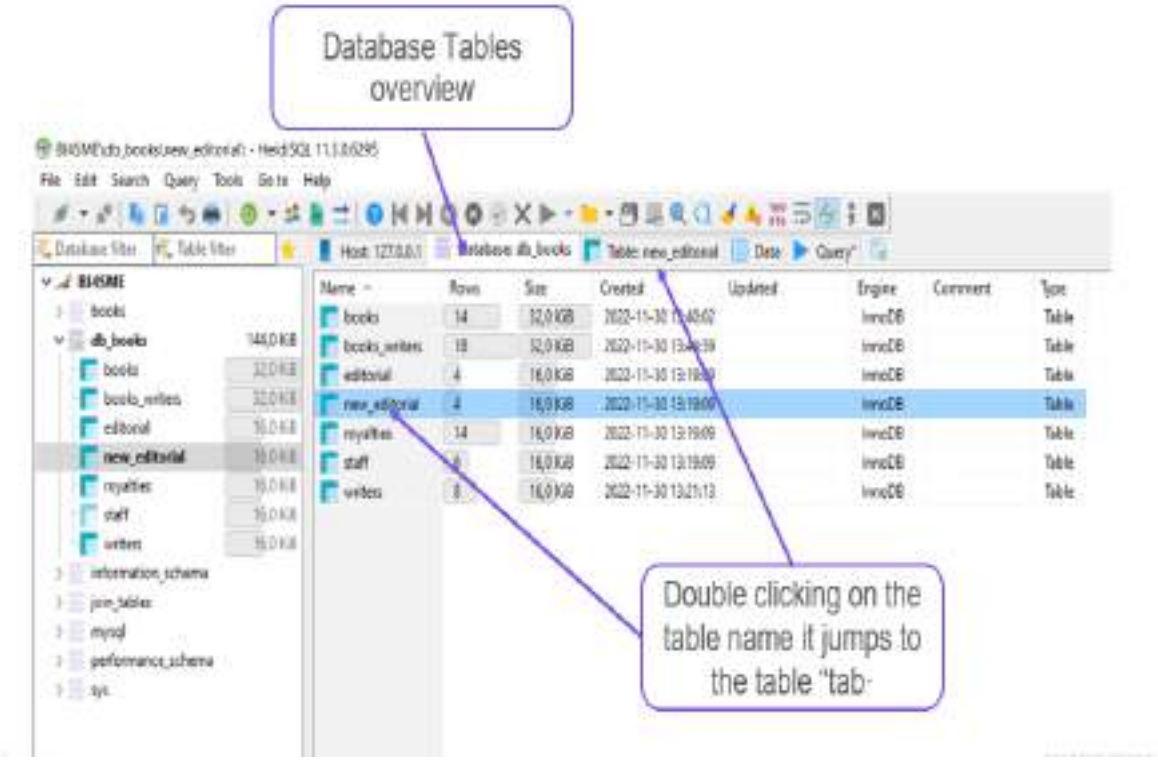
Co-funded by the European Union
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

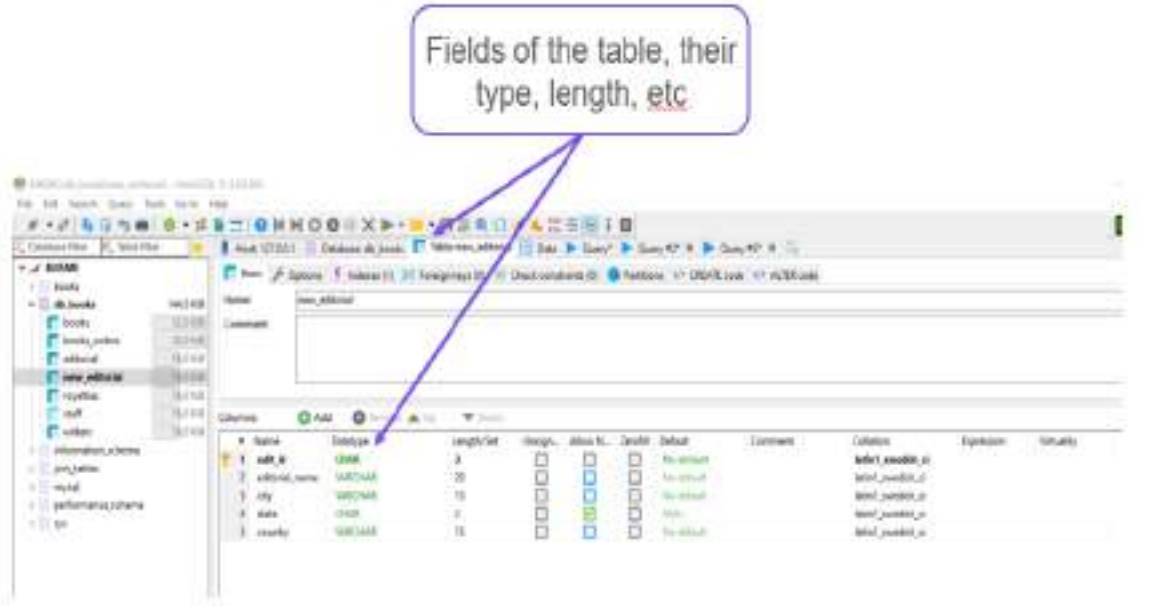
Esta pregunta puede no ser tan fácil de resolver si no estamos familiarizados con el contenido de las tablas de la base de datos y sus campos.
Afortunadamente, estamos utilizando una base de datos muy simple en nuestro curso, pero puedes imaginar la complejidad de una base de datos en producción para una empresa real.

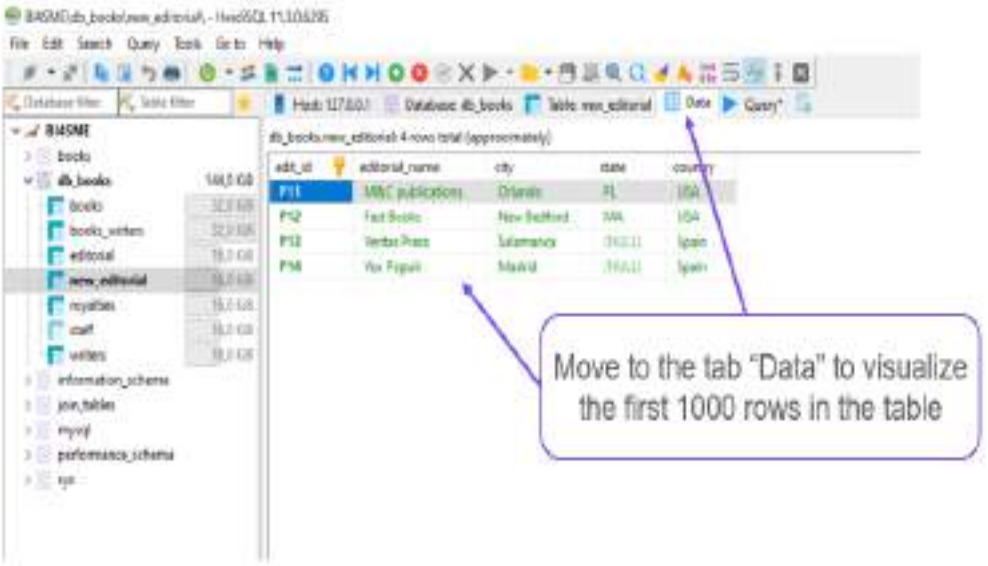
Por esta razón, siempre es una buena práctica tener un plan de la base de datos, donde se describa el contenido de las tablas, la descripción de los campos y la estructura referencial de la base de datos (enlaces entre las tablas). También es altamente recomendable tener un diccionario de datos de la organización donde el analista de datos pueda encontrar fácilmente los campos y su descripción.

No tenemos información tan completa sobre nuestra base de datos de libros, pero Heidi tiene algunas características para explorar los datos, veamos algunas de ellas.

EXPLORACIÓN DE DATOS


Database Tables overview
Double clicking on the table name it jumps to the table "tab-


Fields of the table, their type, length, etc



RESOLVIENDO EL QUERY

Entonces, hemos explorado las tablas de libros en busca de los campos que contienen la información necesaria para nuestro informe.

Recuerda nuestra consulta:

"Obtener una tabla con el título del libro, su tipo y el número de páginas que contiene."

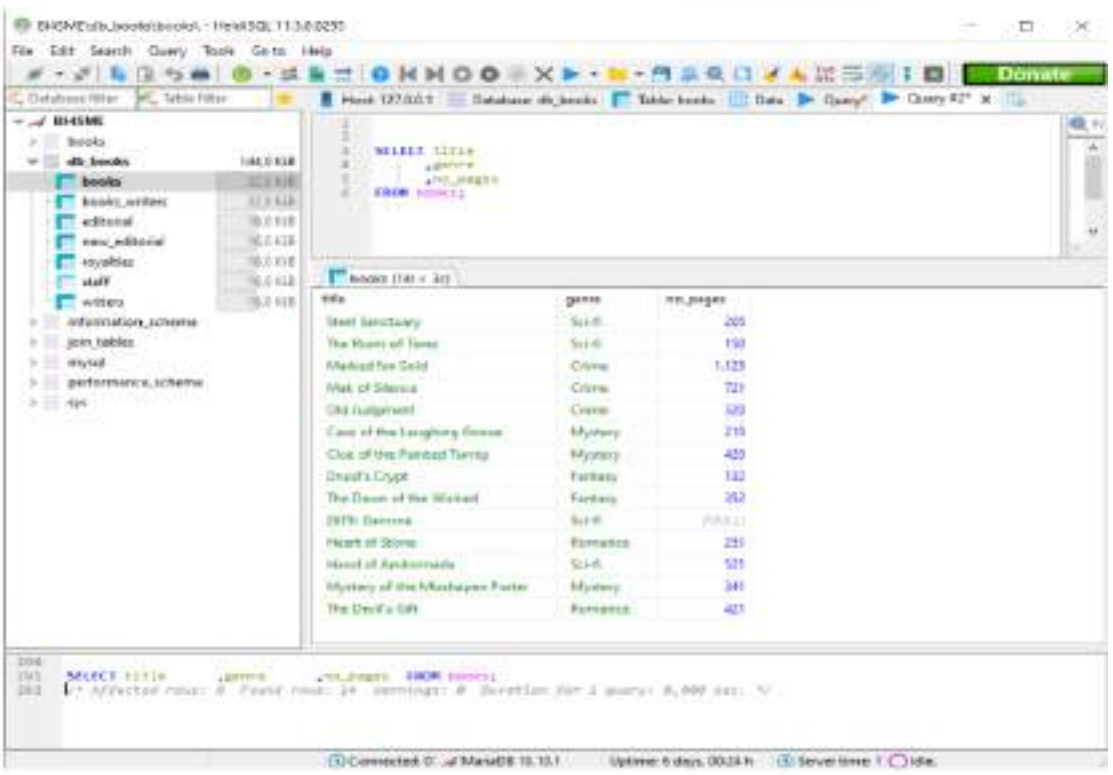
Buscando en las tablas de la base de datos, encontramos la tabla "books" que contiene los campos con los datos necesarios para el informe.

Como los creadores de la base de datos utilizaron un criterio semántico lógico para nombrar las tablas y sus campos, fue fácil deducir la respuesta a las preguntas anteriores:

- En qué base de datos tenemos la información que estamos buscando?
  'db_books'
- Qué campos contienen la información?
  'title', 'genre', 'no_pages'
- Qué responde a la siguiente pregunta, en qué tabla?
  'books'

Sabiendo dónde están disponibles los datos, escribir la consulta es sencillo.

```
SELECT title
      ,genre
      ,no_pages
FROM books;
```



Está bien, pero fíjate cómo el resultado de la consulta utiliza los mismos nombres de columna que los campos en la tabla.

En nuestro informe queremos usar nombres específicos para cada columna:

“Title Name”, “Subject”, “No of pages”

EL ALIAS

En SQL podemos definir el nombre de las columnas en el resultado de la consulta a través de la funcionalidad de alias.

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

- Los alias en SQL se utilizan para dar un nombre temporal a una tabla o a una columna en una tabla.
- Los alias se utilizan a menudo para hacer que los nombres de las columnas sean más legibles.
- Un alias solo existe durante la duración de esa consulta.

Un alias se crea con la palabra clave AS; sin embargo, este comando es opcional y puede omitirse.

Notación para Alias:

```
SELECT column_1 AS 'new_name1'
      , column_2 AS 'new_name2'
      , .....
FROM table AS 'new_table';
```

Title_Name	Subject	No of pages
Steel Sanctuary	Sci-fi	205
The Ruins of Time	Sci-fi	160
Marked for Gold	Crime	1125
Mak of Silence	Crime	721
Old Judgment	Crime	320
Case of the Laughing Goose	Mystery	210
Clue of the Painted Turnip	Mystery	420
Druid's Crypt	Fantasy	132
The Dawn of the Wicked	Fantasy	352
2619: Gamma	Sci-fi	NULL
Heart of Stone	Romance	251
Hand of Andromeda	Sci-fi	521
Mystery of the Misshapen Porter	Mystery	341
The Devil's Gift	Romance	421

2.2.2. El DISTINCT keyword

DISTINCT se usa para eliminar datos duplicados en una consulta. Como resultado, obtenemos solo valores diferentes en las columnas afectadas por la palabra clave DISTINCT.

Notación:

```
SELECT DISTINCT column1, column2, ...
FROM table;
```

UNICIDAD

Si DISTINCT contiene más de una columna, las columnas combinadas determinan la unicidad de las filas.
Veamos un ejemplo en esta tabla.

column1	column2	column3
A	1	C
A	2	B
B	1	C
A	2	B

```
SELECT DISTINCT column1
              , column2
              , column3
FROM table;
```

El query:
SELECT DISTINCT *column1*, *column2, column*
FROM *table*;
DISTINCT considera la fila A, 2, B como duplicada y la incluirá solo una vez en el resultado de la consulta.

column1	column2	column3
A	1	C
A	2	B
B	1	C

EJEMPLOS

"List the state of the writers residence"

Query:

```
SELECT state
FROM writers
;
```

Si resolvemos esta consulta con un simple SELECT, el informe contendrá valores duplicados.

```
+--------+
| state  |
+--------+
| MI     |
| TX     |
| IA     |
| NY     |
| NY     |
| CA     |
| CA     |
| MA     |
+--------+
```

No tiene sentido tener duplicados en la lista.
La alternativa con DISTINCT elimina los duplicados.

```
SELECT DISTINCT state
FROM writers
;
```


Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

```
+-------+
| state |
+-------+
| MI    |
| TX    |
| IA    |
| NY    |
| CA    |
| MA    |
+-------+
```

"Get the unique genre and publisher combinations of the books in our database."

Query:

```
SELECT DISTINCT genre
      , edit_id
FROM books
;
```

```
+---------+---------+
| genre   | edit_id |
+---------+---------+
| Sci-fi  | P01     |
| Sci-fi  | P03     |
| Crime   | P02     |
| Crime   | P04     |
| Mystery | P01     |
| Mystery | P03     |
| Fantasy | P04     |
| Romance | P04     |
+---------+---------+
```

2.2.3. El ORDER BY keyword

ORDER BY SYNTAXIS

Las declaraciones **SELECT** y **FROM por sí solas devuelven datos sin ningún orden**, Para ordenar los datos por columna, utilizamos el comando **ORDER BY**.

La palabra **ORDER BY** siempre está al final de la consulta.

La palabra clave **ORDER BY** ordena los registros en orden ascendente por defecto. Para ordenar los registros en orden descendente, utiliza la palabra clave **DESC**.

Notation:

```
ORDER BY column_1, column_2, .... ASC/DESC;
```

"List the books names and order by its price"

Ascending order

```
SELECT title "Title Name"
      , price
FROM books
ORDER BY price ;
```

Title Name	price
2619: Gamma	NULL
Old Judgment	7.53
Heart of Stone	9.25
The Ruins of Time	12.30
Druid's Crypt	13.20
Hand of Andromeda	16.32
The Dawn of the Wicked	16.32
Case of the Laughing Goose	18.20
Steel Sanctuary	21.30
Mak of Silence	24.30
Clue of the Painted Turnip	25.00
The Devil's Gift	29.23
Marked for Gold	32.10
Mystery of the Misshapen Porter	35.60

Descending order

```
SELECT title "Title Name"
      , price
FROM books
ORDER BY price DESC;
```

Title Name	price
Mystery of the Misshapen Porter	35.60
Marked for Gold	32.10
The Devil's Gift	29.23
Clue of the Painted Turnip	25.00
Mak of Silence	24.30
Steel Sanctuary	21.30
Case of the Laughing Goose	18.20
The Dawn of the Wicked	16.32
Hand of Andromeda	16.32
Druid's Crypt	13.20
The Ruins of Time	12.30
Heart of Stone	9.25
Old Judgment	7.53
2619: Gamma	NULL

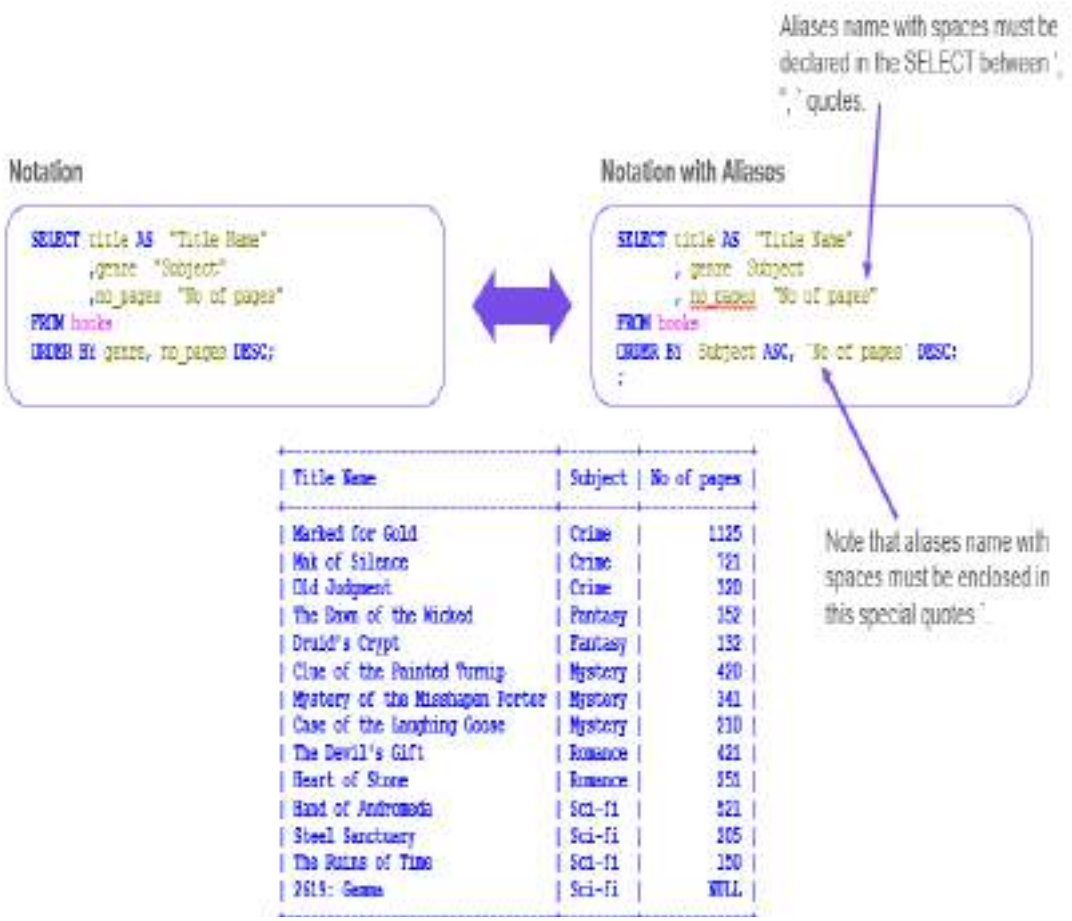
ORDER BY uso de ALIASES

"Sort our previous report by topic in ascending order and then by the number of pages in descending order".

Alias se pueden specificar en SORT columns.

Se pueden especificar alias en lugar de las columnas de ordenación. Pero la notación puede ser un poco complicada cuando nos encontramos con nombres de alias que incluyen espacios, por ejemplo, "No of pages". En este caso, debemos usar este tipo de comillas ` para deformar el nombre del alias en las declaraciones ORDER BY.

Ejemplo:



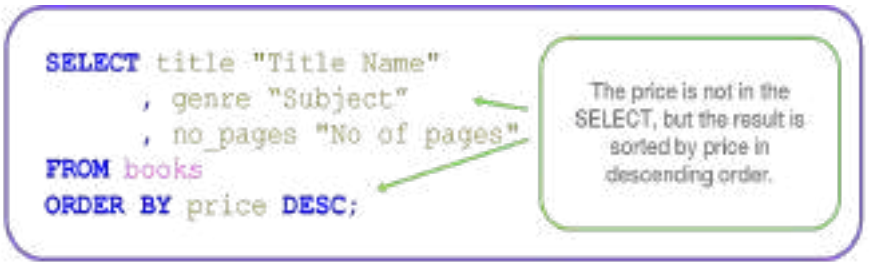
ORDER BY – WITH NOT SELECTED FIELDS

SORT se puede hacer para columnas que no están en la cláusula SELECT.

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

"List the title names, subject, no of pages and price, sort the result from highest to lowest price."



Title Name	Subject	No of pages
Mystery of the Misshapen Porter	Mystery	341
Marked for Gold	Crime	1125
The Devil's Gift	Romance	421
Clue of the Painted Turnip	Mystery	420
Mak of Silence	Crime	721
Steel Sanctuary	Sci-fi	205
Case of the Laughing Goose	Mystery	210
The Dawn of the Wicked	Fantasy	352
Hand of Andromeda	Sci-fi	521
Druid's Crypt	Fantasy	132
The Ruins of Time	Sci-fi	150
Heart of Stone	Romance	251
Old Judgment	Crime	320
2619: Gamma	Sci-fi	NULL

"Reproduce the above report but the price should not be visible in the table".

Adding the column price

```
SELECT title "Title Name"
      , genre "Subject"
      , no_pages "No of pages"
      , price
FROM books
ORDER BY price DESC;
```

Title Name	Subject	No of pages	price
Mystery of the Misshapen Porter	Mystery	341	35.60
Marked for Gold	Crime	1125	32.10
The Devil's Gift	Romance	421	29.23
Clue of the Painted Turnip	Mystery	420	25.00
Mak of Silence	Crime	721	24.30
Steel Sanctuary	Sci-fi	205	21.30
Case of the Laughing Goose	Mystery	210	18.20
The Dawn of the Wicked	Fantasy	352	16.32
Hand of Andromeda	Sci-fi	521	16.32
Druid's Crypt	Fantasy	132	13.20
The Ruins of Time	Sci-fi	150	12.30
Heart of Stone	Romance	251	9.25
Old Judgment	Crime	320	7.53
2619: Gamma	Sci-fi	NULL	NULL

Solo verifica cómo ambas tablas tienen un orden de filas idéntico.

2.2.4. El WHERE keyword

INTRODUCCIÓN A WHERE

Si estás familiarizado con Excel, sabrás lo común que es filtrar las filas de una tabla por alguna condición. Así, no mostramos todos los registros (filas) de la tabla, sino solo aquellos en los que uno de los campos (columnas) cumple una determinada condición.

Los filtros en SQL se realizan con la palabra clave WHERE.

Se usa para extraer solo aquellos registros que cumplen una condición especificada.

Notación:

WHERE condition_1
AND/OR condition_2

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

AND/OR conditon_3...

AND/OR son opcionales; utilizamos estas palabras clave para vincular condiciones, lo veremos con ejemplos a continuación.

La cláusula WHERE no solo se usa en declaraciones SELECT (consultas), ¡sino también en UPDATE, DELETE, etc.!

Aquí hay una tabla de los operadores para expresar condiciones en SQL:

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

PRIMEROS QUERIES CON WHERE

Vamos a usar algunos de estos operadores con unos ejemplos!

"List authors whose surname is not FRAN"

```
SELECT wr_fname AS "name"
     , wr_lname AS "Last name"
FROM writers
WHERE wr_fname <> 'Fran';
```

name	Last name
Randolf	Stormy
Sam	Christobel
Charlene	Osborne
Alene	Loreen
	Romuald
Emma	Tony
Bruno	Randolph

"List books without a signed contract"

```
SELECT title
  , genre
  , contract
FROM books
WHERE contract = 0;
```

title	genre	contract
2619: Gamma	Sci-fi	0

"List titles published since 2001"

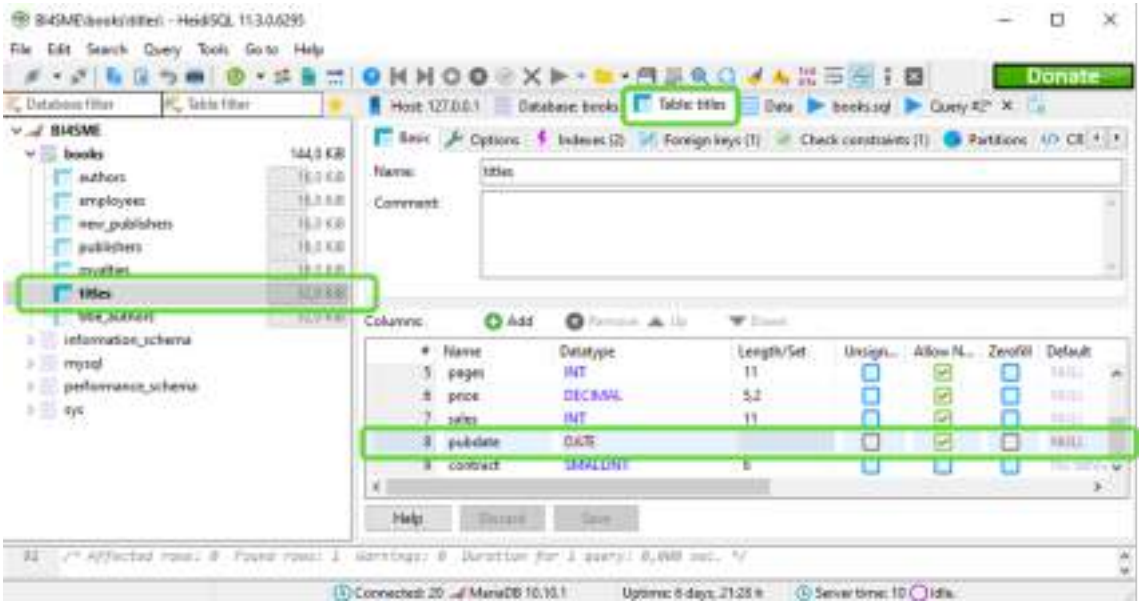
Aquí tenemos nuestra primera condición sobre campos de fecha. **Las consultas que involucran fechas pueden** ser un poco complicadas. Es

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

importante asegurarse de que el motor de búsqueda SQL pueda reconocer la cadena como una fecha. Este tema se tratará en un capítulo posterior. Por ahora, solo necesitamos una exploración de datos simple.

1. Selecciona la tabla books y ve a la pestaña "Table: titles".
2. Allí verás una lista de las características de la tabla y su tipo.
3. El campo "pubdate" tiene el tipo de dato "DATE", que es el tipo correcto para un campo que contiene fechas.



Ok, eso parece prometedor! Los campos parecen estar formateados correctamente y podemos entender las condiciones en las fechas.

Si escribimos una condición como esta:

pubdate < '2001-01-01'

El query tomará todos los registros hasta el primero de enero de 2001. Esto es posible porque el motor SQL entiende la fecha que estamos escribiendo y puede aplicar la condición correctamente.

¡Vamos a escribir el query!

```
SELECT title, pubdate
FROM books
WHERE pubdate >= '2001-01-01';
```

```
+---------------------------------+------------+
| title                           | pubdate    |
+---------------------------------+------------+
| The Ruins of Time               | 2001-04-15 |
| Marked for Gold                 | 2001-05-20 |
| Mak of Silence                  | 2010-06-12 |
| Druid's Crypt                   | 2020-01-17 |
| The Dawn of the Wicked          | 2020-04-16 |
| Hand of Andromeda               | 2001-08-17 |
| Mystery of the Misahagen Porter | 2015-07-28 |
| The Devil's Gift                | 2014-09-26 |
+---------------------------------+------------+
```

Combinación de condiciones con lógica AND, OR, NOT

La condición a utilizar en la cláusula WHERE puede ser tan compleja como puedas imaginar, encadenando o anidando condiciones utilizando los operadores condicionales AND, OR, NOT.

El funcionamiento lógico de estos operadores se describe en esta tabla. Para AND y OR presentamos una matriz de doble entrada para dos condiciones, pero puedes encadenar tantas condiciones como sea necesario.

AND Logic Matrix

AND	Second condition		
First condition	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

Entendamos la lógica!
Si la primera condición es TRUE (VERDADERO) y la segunda es FALSA, el resultado es FALSO.

AND	Second condition		
First condition	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

Ahora, veamos un ejemplo!

"List Sci-fi books priced under 20 EUR"

```
SELECT title, genre, price
FROM books
WHERE genre = 'Sci-fi' AND price < 20;
```

```
+-------------------+--------+-------+
| title             | genre  | price |
+-------------------+--------+-------+
| The Ruins of Time | Sci-fi | 12.30 |
| Hand of Andromeda | Sci-fi | 16.32 |
+-------------------+--------+-------+
```

TRUE + TRUE = TRUE
(the row is included in the result)

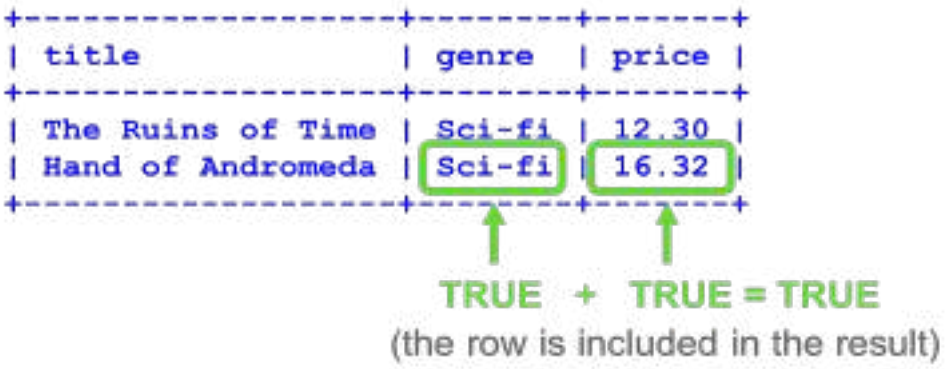
OR Logic Matrix

OR	Second condition		
First condition	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	FALSE
NULL	TRUE	NULL	NULL

Observa la diferencia entre estas sentencias:

"Lista de libros de ciencia ficción con un precio inferior a 20 EUR": Un libro se incluirá en el resultado si su tipo es "Sci_fi" y su precio es inferior a 20 EUR.

"Lista de libros de ciencia ficción o los libros con un precio inferior a 20 EUR": Los títulos se seleccionan si pertenecen a la categoría de ciencia ficción o si tienen un precio inferior a 20 EUR, independientemente de su tipo.



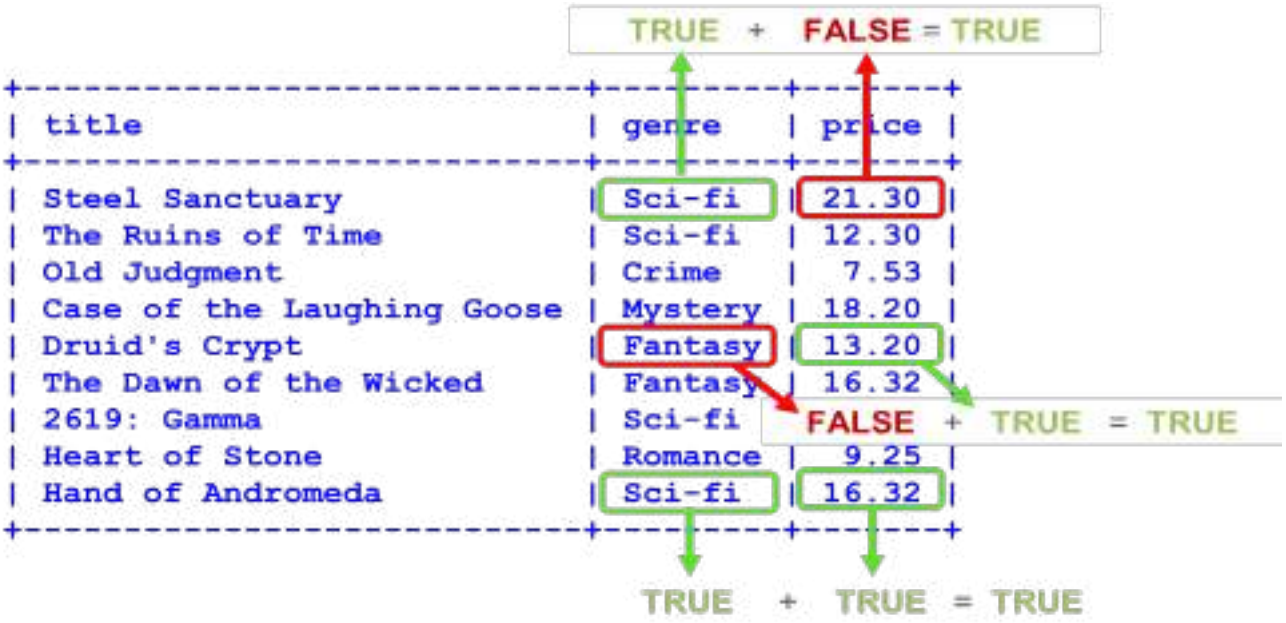
Ahora podemos ir con un ejemplo de lógica OR.

"List Sci-fi books or the books priced under 20 EUR"

```
SELECT title, genre, price
FROM books
WHERE genre = 'Sci-fi' OR price < 20;
```


Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132



NOT Logic Matrix

NOT	
Condition	Result
TRUE	FALSE
FALSE	TRUE
NULL	TRUE

Y ahora veamos un ejemplo

"Vamos a obtener una lista de autores que no residen en California."
Hay dos soluciones potenciales para esta consulta.

Filtrar por el campo state_of_residence no igual (<>) a “CA”.
Filtrar por campos iguales (=) a "CA" y luego invertir el resultado con NOT.

Co-funded by the European Union
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

```
SELECT wr_fname AS "name"
     , wr_lname AS "Last name"
FROM writers
WHERE wr_fname <> 'Fran';
```

name	Last name
Randolf	Stormy
Sam	Christobel
Charlene	Osborne
Alene	Loreen
	Romuald
Emma	Tony
Bruno	Randolph

```
SELECT wr_fname AS "name"
     , wr_lname AS "Last name"
FROM writers
WHERE NOT wr_fname = 'Fran';
```

name	Last name
Randolf	Stormy
Sam	Christobel
Charlene	Osborne
Alene	Loreen
	Romuald
Emma	Tony
Bruno	Randolph

Usar la palabra clave **NOT** delante de una condición invierte el resultado de la condición del filtro. Si el filtro incluye múltiples condiciones, podemos usar paréntesis para agrupar la condición después de la condición **NOT.**

Ve el ejemplo a continuación.

"List Sci-fi books under 20 EUR"

```
SELECT title, genre, price
FROM books
WHERE genre = 'Sci-fi' AND price < 20;
```

title	genre	price
The Ruins of Time	Sci-fi	12.30
Hand of Andromeda	Sci-fi	16.32

"No Sci-fi books over 20 EUR"

```
SELECT title, genre, price
FROM books
WHERE NOT(genre = 'Sci-fi' AND price < 20);
```

title	genre	price
Steel Sanctuary	Sci-fi	21.30
Marked for Gold	Crime	32.13
Mak of Silence	Crime	24.30
Old Judgment	Crime	7.53
Case of the Laughing Goose	Mystery	19.25
Clue of the Painted Turnip	Mystery	25.00
Druid's Crypt	Fantasy	13.20
The Dawn of the Wicked	Fantasy	16.32
Heart of Stone	Romance	9.25
Mystery of the Misshapen Potter	Mystery	35.63
The Devil's Gift	Romance	29.23

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

2.2.5. El WHERE_BETWEEN keyword

FILTROS DE RANGO BETWEEN

Podemos seleccionar valores que se encuentran entre un valor superior e inferior con los comandos condicionales que ya hemos aprendido.

Uso del filtrado de rango con condiciones encadenadas con AND.

Notación:

```
WHERE column<=higher_value
      AND
        column>= lower_value
```

O Podemos usar el comando **BETWEEN**.

```
WHERE column BETWEEN lower_value
                 AND higher_value
```

- BETWEEN funciona con caracteres, números y fechas.

- El rango de BETWEEN contiene un valor inferior y un valor superior separados por AND. El valor inferior debe ser menor o igual al valor superior.
- El rango es inclusivo, incluye los valores inferiores y superiores en el resultado.

- Una condición BETWEEN puede invertirse con **NOT.**

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

Ejemplo:

"List titles published in 2001"

Podemos abordar esta consulta de dos maneras: usando operadores condicionales o BETWEEN.

Conditional operators

```
SELECT title, pubdate
FROM books
WHERE pubdate >='2001-01-01'
   AND pubdate <='2001-12-31';
```

```
+-------------------+------------+
| title             | pubdate    |
+-------------------+------------+
| The Ruins of Time | 2001-04-15 |
| Marked for Gold   | 2001-05-20 |
| Hand of Andromeda | 2001-08-17 |
+-------------------+------------+
```

Between

```
SELECT title, pubdate
FROM books
WHERE pubdate
   BETWEEN '2001-01-01'
       AND '2001-12-31';
```

```
+-------------------+------------+
| title             | pubdate    |
+-------------------+------------+
| The Ruins of Time | 2001-04-15 |
| Marked for Gold   | 2001-05-20 |
| Hand of Andromeda | 2001-08-17 |
+-------------------+------------+
```

"List titles NOT published in 2001"

Una condición BETWEEN puede invertirse con **NOT.**

```
SELECT title, pubdate
FROM books
WHERE pubdate NOT BETWEEN '2001-01-01' AND '2001-12-31';
```


Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

title	pubdate
Steel Sanctuary	1998-05-12
Mak of Silence	2010-06-12
Old Judgment	2000-06-02
Case of the Laughing Goose	1995-07-26
Clue of the Painted Turnip	1995-08-04
Druid's Crypt	2020-01-17
The Dawn of the Wicked	2020-04-16
Heart of Stone	1992-01-18
Mystery of the Misshapen Porter	2015-07-28
The Devil's Gift	2014-09-26

2.2.6. El WHERE_LIKE keyword

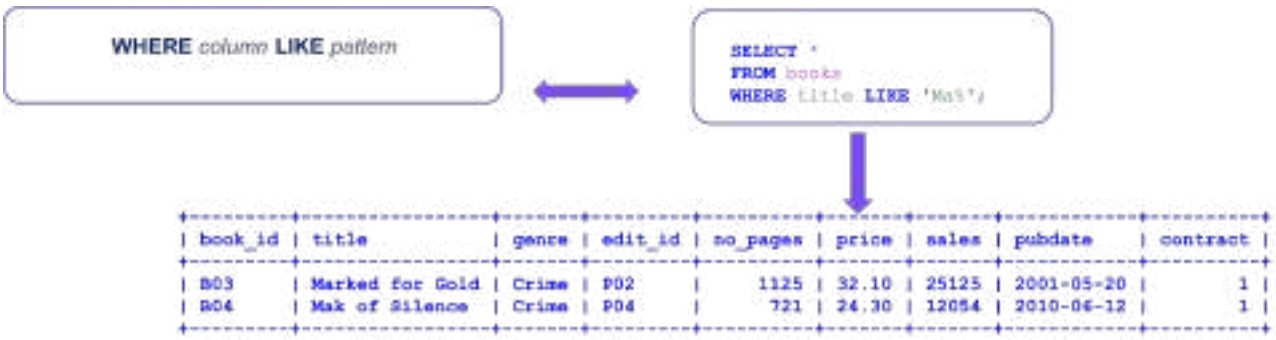
El comando **LIKE** se utiliza para encontrar una **coincidencia con un patrón**. Por ejemplo, todos los libros cuyo título comienza con la palabra "Not" o todos los libros cuyo título contiene la palabra "System".

- **LIKE** funciona solo con caracteres, no con números o fechas.
- **LIKE** usa un patrón para coincidir.
- El carácter '%' coincide con una cadena con cero o más caracteres.
- El carácter '_' (guion bajo) coincide con un solo carácter.
- **NOT** puede preceder a **LIKE**, lo que invierte el resultado.

Los patrones pueden ser tan complejos como puedas imaginar. Aquí hay algunos de los más comunes y fáciles de usar.

Pattern	Matching
A%'	String of size>=1 starting with 'A' Example: 'America', 'Anonimous'.
%s'	String of size>=1 ending in 's' Example: 'Cats', 'Anonimous'.
%im%'	String of size>=2 containing anywhere "im", Example: "impossible", "anonymous
____'	Any 4-character string, e.g. 'village', 'town', 'city', 'dogs'.
St__'	A 5-character string starting with 'St' and followed by 3 characters. Example Style, Start
__re%'	String with size>=3 ,starts with any character followed by 're'.
_re_'	4 character string with "re" as the second and third character
%re_'	String of size >= 3 whose penultimate and antepenultimate characters are re. Examples:

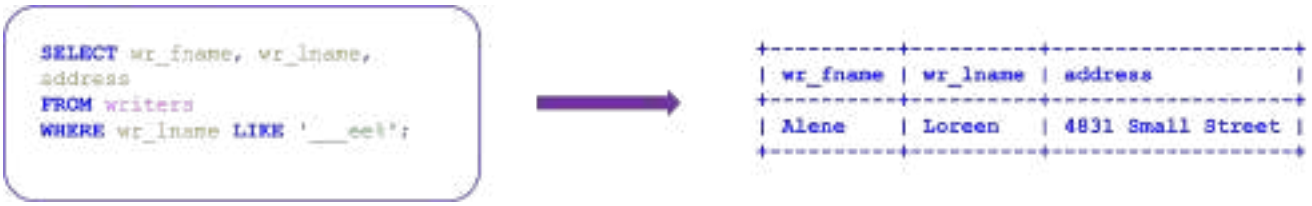
"List books whose title begins with 'Ma' "



Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

"List of authors whose surnames have 'ee' in the fourth and fifth positions".



2.2.7. El WHERE_IN keyword

FILTRAR POR VALORES ALMACENADOS EN UNA LISTA CON "IN"

Si queremos filtrar por diferentes valores, ya conocemos la opción de usar la palabra clave OR en una declaración WHERE.

"Seleccionar libros publicados por las editoriales P01 y P02"

```
SELECT title, genre, edit_id
FROM books
WHERE edit_id='P01'
        OR
         edit_id='P02';
```

title	genre	edit_id
Steel Sanctuary	Sci-fi	P01
Marked for Gold	Crime	P02
Case of the Laughing Goose	Mystery	P01
2619: Gamma	Sci-fi	P01
Hand of Andromeda	Sci-fi	P01

Sin embargo, esta notación puede ser engorrosa si el número de condiciones a filtrar es grande.

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

En este caso, el uso de la notación **IN** en la cláusula WHERE es el enfoque correcto.

Notation

```
WHERE column IN (value_1, value_2, Value_3,….)
```

Algunos comentarios sobre IN:

- Los valores en la lista **IN** pueden ser cadenas, números o fechas.
- El orden en la lista no es relevante.
- Puede invertirse con la palabra clave **NOT**.

Veamos algunos ejemplos!

"Select books published by editorial P01 or P02"

Aplicando la notación IN a la consulta anterior.

```
SELECT title, genre, edit_id
FROM books
WHERE edit_id IN ('P01' , 'P02');
```

title	genre	edit_id
Steel Sanctuary	Sci-fi	P01
Marked for Gold	Crime	P02
Case of the Laughing Goose	Mystery	P01
2619: Gamma	Sci-fi	P01
Hand of Andromeda	Sci-fi	P01

"List of writers living in New York or Los Angeles"

```
SELECT wr_fname AS 'first_name'
      ,wr_lname AS 'last_name'
      , city
FROM writers
WHERE city IN ('New York' , 'Los Angeles');
```

```
+------------+-----------+-------------+
| first_name | last_name | city        |
+------------+-----------+-------------+
| Alene      | Loreen    | New York    |
|            | Romuald   | Los Angeles |
| Emma       | Tony      | Los Angeles |
+------------+-----------+-------------+
```

"List of writers who do not live in New York or Los Angeles"

```
SELECT wr_fname AS 'first_name'
     , wr_lname AS 'last_name'
     , city
FROM writers
WHERE city
NOT IN ('New York' , 'Los Angeles')
;
```

first_name	last_name	city
Randolf	Stormy	Grand Rapids
Sam	Christobel	College Station
Charlene	Osborne	Sergeant Bluff
Ryker	Fran	Buffalo
Bruno	Randolph	Boston

2.2.8. El NULL dilemma

MANEJO DE VALORES NULL

Manejar valores nulos en SQL siempre es un poco complicado. **Los valores nulos se refieren a valores** desconocidos y las palabras clave utilizadas para filtrar en la cláusula WHERE, como BETWEEN, LIKE, IN... no pueden encontrar valores NULL.

Veamos esto con algunos ejemplos.
Hay algunos valores NULL en la tabla editorial.

Ejemplo:

edit_id	editorial_name	city	state	country
P01	Mattson and Cooper	Anaheim	CA	USA
P02	Press News	Detroit	MI	USA
P03	Pegasus Pub	Paris	NULL	France
P04	Dorrance Publishing	Paris	NULL	France

Veamos qué pasa si busco: *"Publishers not located in California"*

```
SELECT editorial_name
     , state
FROM editorial
WHERE state <> 'CA'
```

```
+----------------+-------+
| editorial_name | state |
+----------------+-------+
| Press News     | MI    |
+----------------+-------+
```

El motor de búsqueda de SQL no sabe cómo interpretar el valor NULL. Dado que el valor es desconocido, SQL no puede asegurar si el valor cumple o no con la condición.

Por lo tanto, los editores con estado NULL no se consideran estrictamente no localizados en California y, por lo tanto, no se incluyen en el listado.

La misma lógica se aplica si buscamos: *"Californian publishers":*

```
SELECT editorial_name
     , state
FROM editorial
WHERE state = 'CA';
```

```
+--------------------+-------+
| editorial_name     | state |
+--------------------+-------+
| Mattson and Cooper | CA    |
+--------------------+-------+
```

INCLUYENDO VALORES NULOS

Si queremos considerar los valores NULL como no californianos, la descripción correcta de la consulta debería ser:
"List publishers not located in California or whose state is unknown (null)"

```
SELECT editorial_name
       , state
FROM editorial
WHERE state = 'CA'
       OR
      state IS NULL
```

```
+-----------------------+-------+
| editorial_name        | state |
+-----------------------+-------+
| Mattson and Cooper    | CA    |
| Pegasus Pub           | NULL  |
| Dorrance Publishing   | NULL  |
+-----------------------+-------+
```

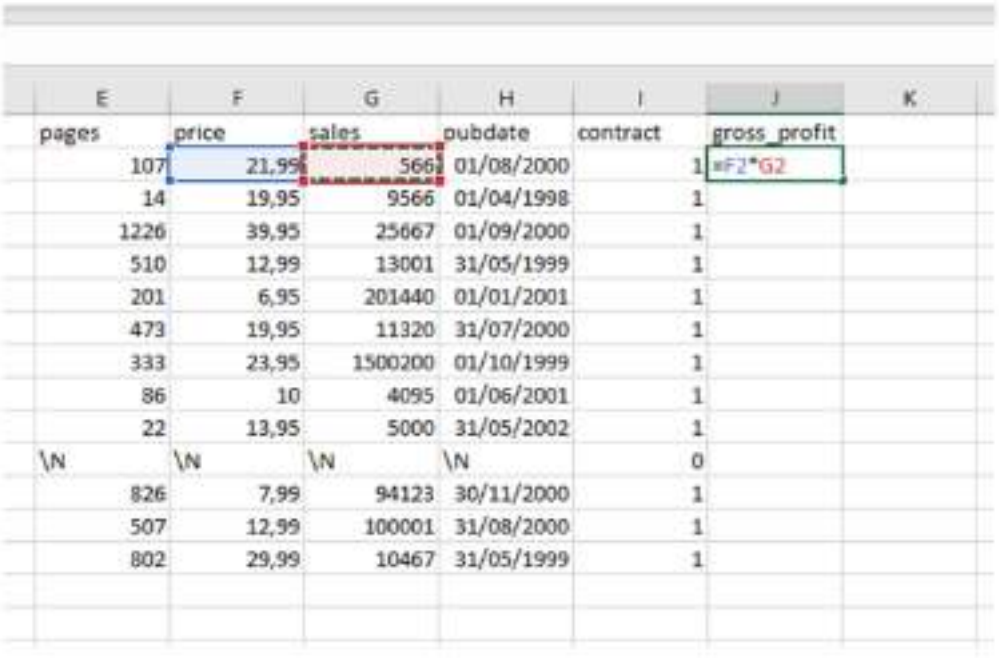
2.2.9. El calculated FIELDS

En las tablas de Excel, es común crear una nueva columna como cálculo aritmético entre una o más columnas.

Veámoslo con algunos ejemplos.

Para añadir una nueva columna en la tabla de archivo de Excel **'titles'** es fácil.

Calculemos el volumen total como la multiplicación de las ventas por el precio.



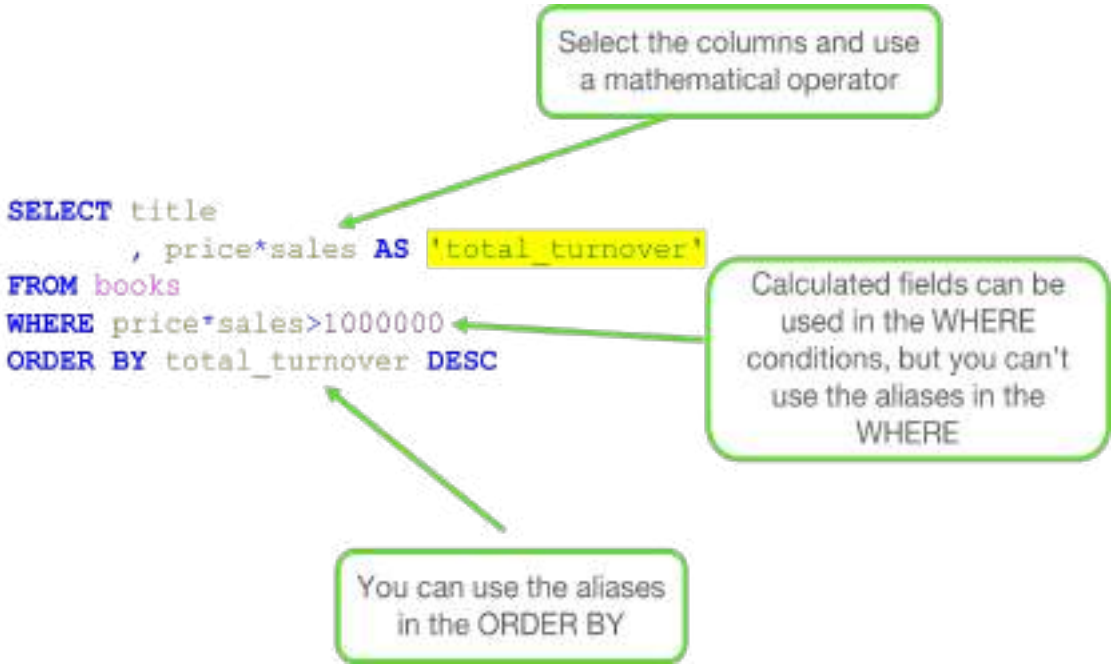
	E	F	G	H	I	J	K
	pages	price	sales	pubdate	contract	gross_profit	
	107	21,99	566	01/08/2000	1	=F2*G2	
	14	19,95	9566	01/04/1998	1		
	1226	39,95	25667	01/09/2000	1		
	510	12,99	13001	31/05/1999	1		
	201	6,95	201440	01/01/2001	1		
	473	19,95	11320	31/07/2000	1		
	333	23,95	1500200	01/10/1999	1		
	86	10	4095	01/06/2001	1		
	22	13,95	5000	31/05/2002	1		
	\N	\N	\N	\N	0		
	826	7,99	94123	30/11/2000	1		
	507	12,99	100001	31/08/2000	1		
	802	29,99	10467	31/05/1999	1		

En SQL, un campo calculado es una nueva columna que se crea como resultado de una operación sobre datos existentes en tablas. Por lo tanto, el nuevo campo no existe en las tablas originales de la base de datos; su información es efímera, ya que solo se presenta en el resultado de la consulta, pero no se almacena en la base de datos.

El SQL CALCULATED FIELDS

Mira esta solicitud de informe: **"Lista de títulos que han generado más de $1,000,000 en volumen de ventas".**

Uhm! Aquí tenemos un problema, en la tabla "titles" no hay un campo que incluya el volumen de ventas, ni en ninguna otra tabla tampoco. Entonces, si pudiéramos agregar una nueva columna calculando el nuevo campo "total_turnover", podríamos resolver el problema.



title	total_turnover
Clue of the Painted Turnip	42246350.00
Hand of Andromeda	32649612.48
Old Judgment	7680788.25
The Devil's Gift	2353628.83

Tabla de OPERADORES

Los operadores matemáticos son los mismos que se utilizan en cualquier expresión matemática básica.

Operator	Meaning	Operates on
+ (Add)	Addition	Numeric value
- (Subtract)	Subtraction	Numeric value
* (Multiply)	Multiplication	Numeric value
/ (Divide)	Division	Numeric value
% (Modulo)	Returns the integer remainder of a division. For example, 17 % 5 = 2 because the remainder of 17 divided by 5 is 2.	Numeric value

Ahora, algunos ejemplos!

"List the books price but apply a discount of 10%"

```
SELECT title
     , price AS 'Old price'
     , price*(1-0.1) AS 'New price'
FROM books
ORDER BY price DESC;
```

title	Old Price	New price
Mystery of the Misshapen Porter	35.60	32.040
Marked for Gold	32.10	28.890
The Devil's Gift	29.23	26.307
Clue of the Painted Turnip	25.00	22.500
Mak of Silence	24.30	21.870
Steel Sanctuary	21.30	19.170
Case of the Laughing Goose	18.20	16.380
The Dawn of the Wicked	16.32	14.688
Hand of Andromeda	16.32	14.688
Druid's Crypt	13.20	11.880
The Ruins of Time	12.30	11.070
Heart of Stone	9.25	8.325
Old Judgment	7.53	6.777
2619: Gamma	NULL	NULL

2.2.10. El GROUP BY keyword

EMPECEMOS CON EXCEL!

Es muy común en la elaboración de informes de datos obtener un agrupamiento de datos agregados de una o más columnas.

Los ejemplos podrían ser:

- el promedio de cada producto diferente,
- el número de estudiantes por asignatura,
- contar el número de personas en una tabla, etc.

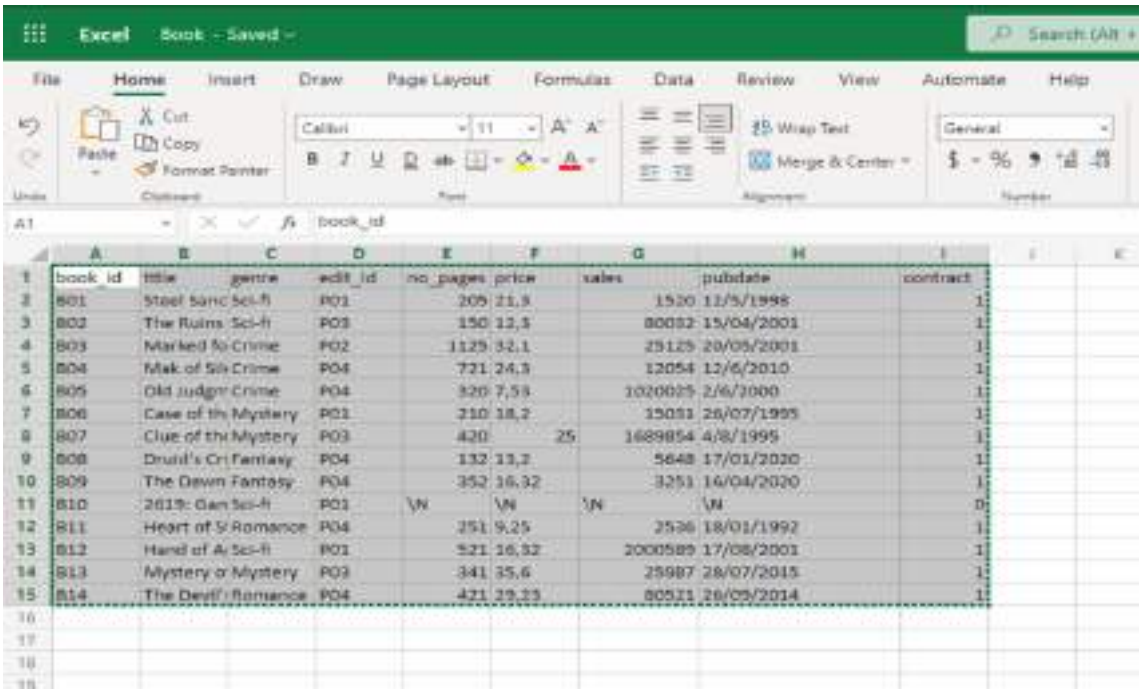
Si estás familiarizado con las tablas dinámicas de Excel, probablemente estás acostumbrado a trabajar con datos agrupados.

Para comprender mejor el concepto de agrupamiento de datos, comenzaremos este capítulo **revisando los conceptos básicos** de las tablas dinámicas en Excel, utilizando las tablas de nuestra base de datos "db_books".

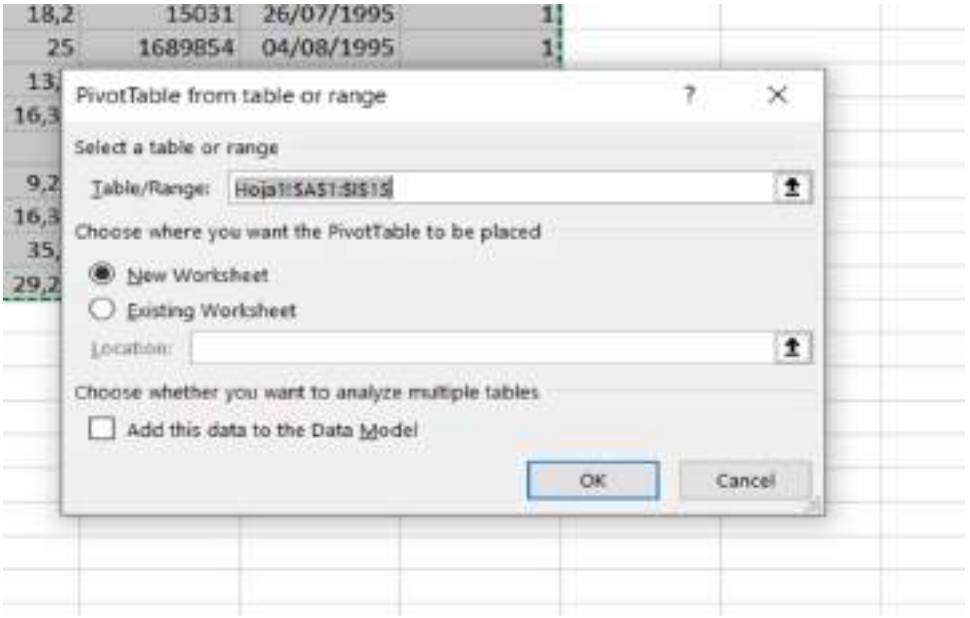
Abramos el archivo de Excel books_tables.xlsx.

Ve a la tabla "titles" en el archivo de Excel; verás los mismos registros que están registrados en la base de datos de libros en la tabla "titles".

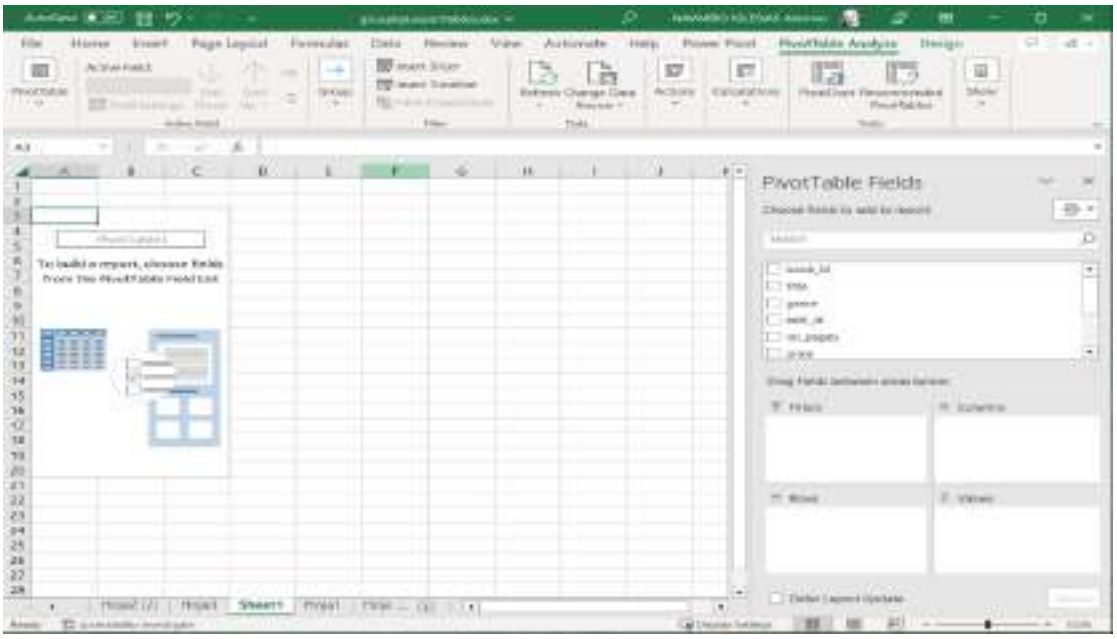
Simplemente selecciona toda la tabla de datos:



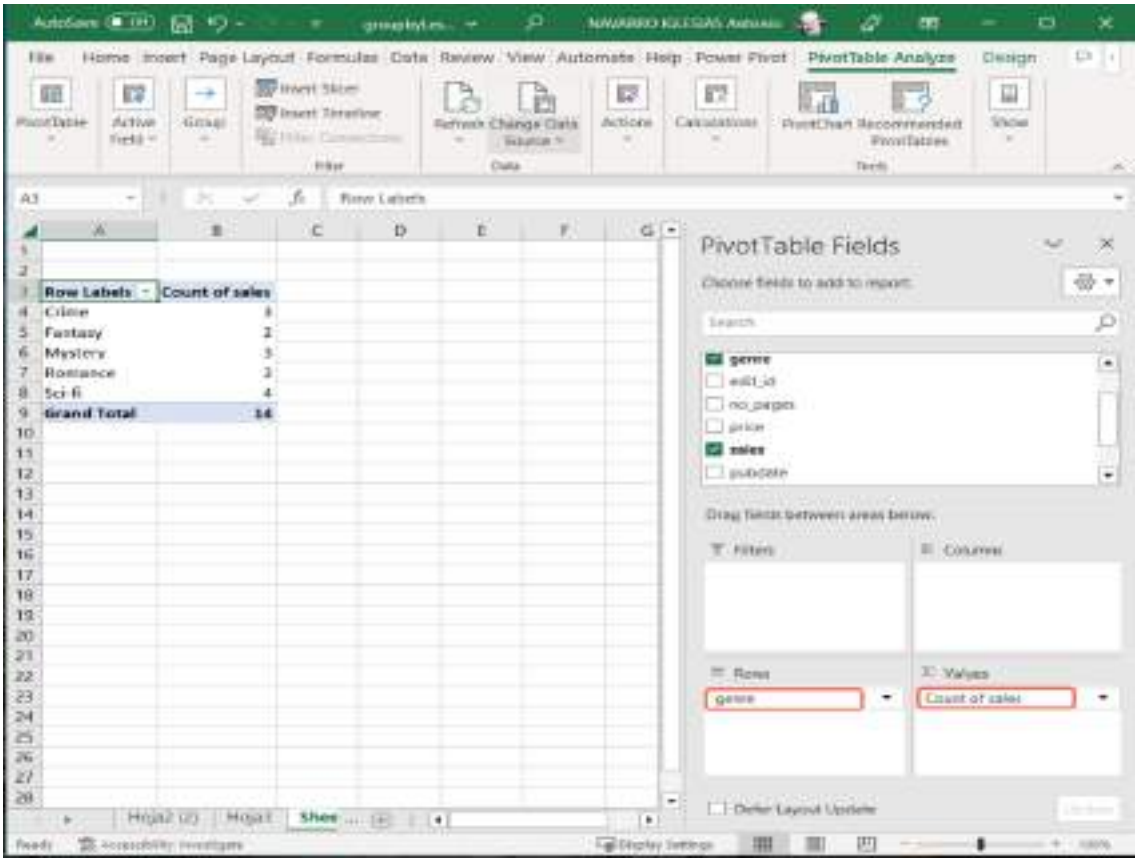
Ve a Insertar y selecciona Tabla dinámica. Haz clic en Aceptar.



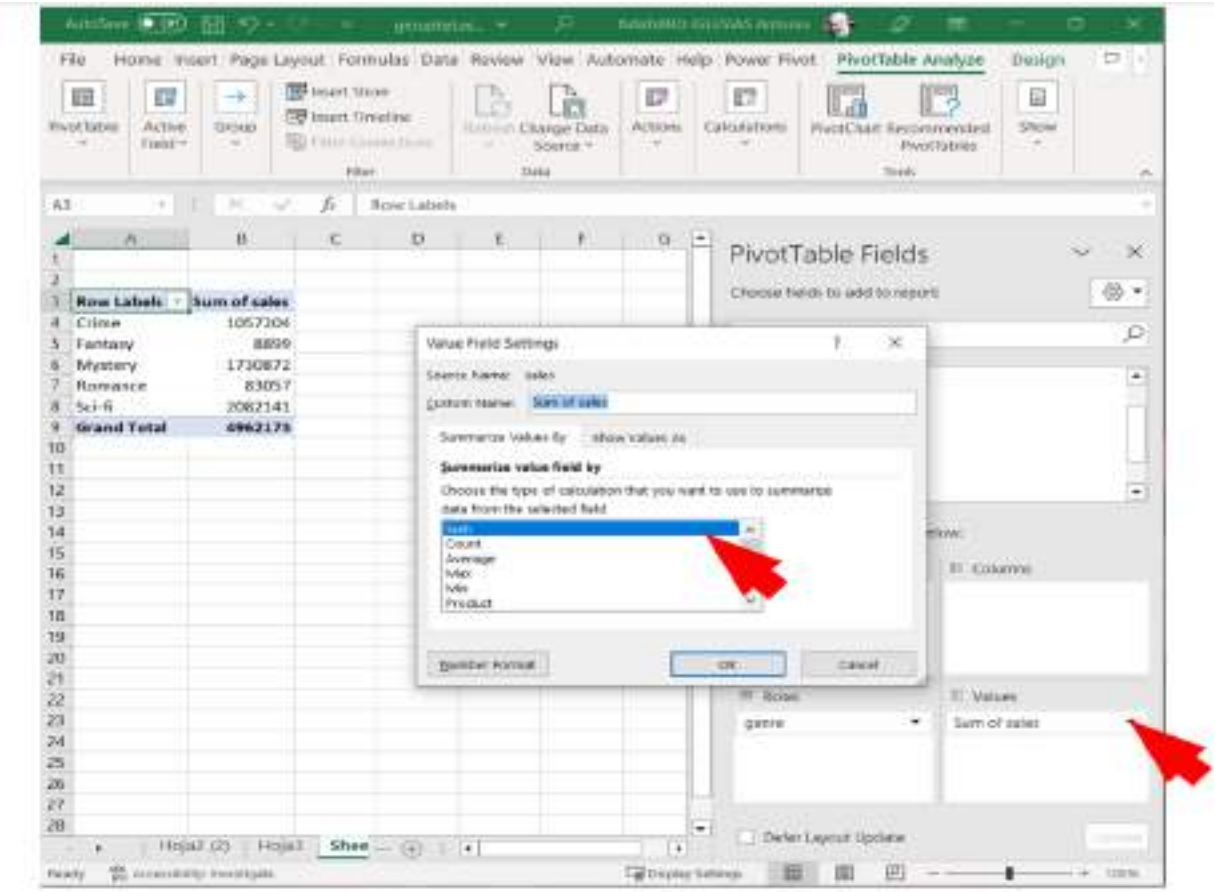
Se abre una nueva hoja de Excel para crear la tabla dinámica.



En el área de selección **fields**, simplemente añade el campo "type" en filas y "sales" en valores.



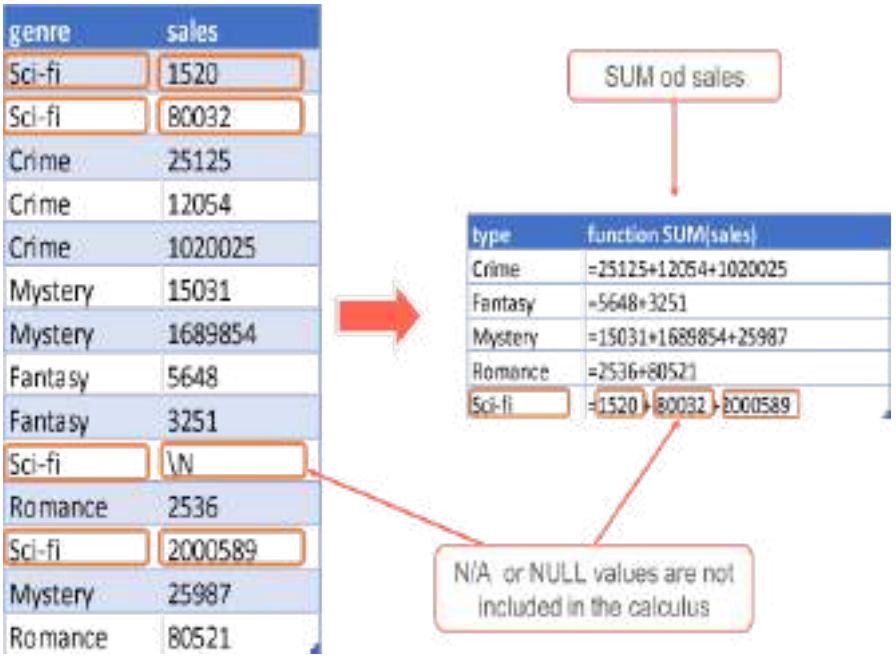
Asegúrate de seleccionar la métrica **Sum** para el KPI agregado.



El resultado es la suma de ventas por tipo de libro.

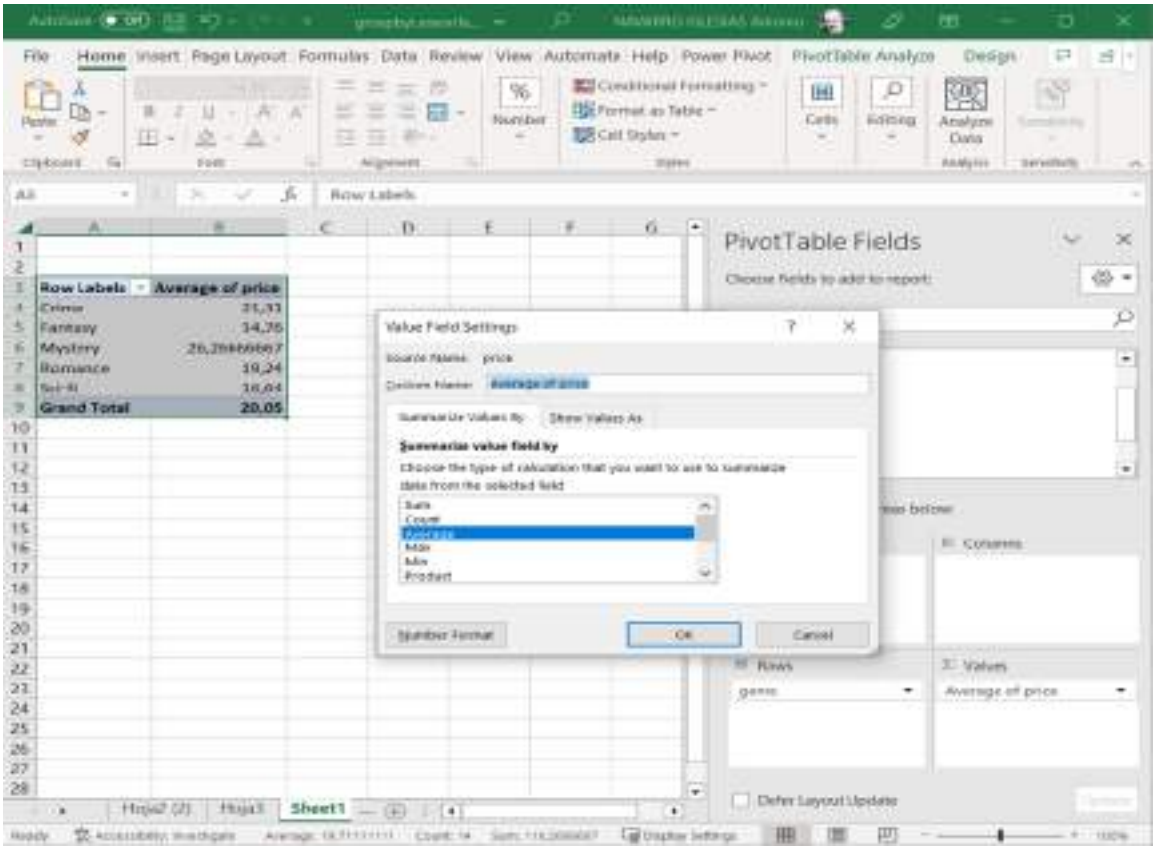
Row Labels	Sum of sales
Crime	1057204
Fantasy	8899
Mystery	1730872
Romance	83057
Sci-fi	2082141
Grand Total	**4962173**

Esta operación interna la realiza la tabla dinámica sumando las ventas para aquellos valores donde el tipo de publicación es el mismo.



OBTÉN EL PRECIO PROMEDIO CON EXCEL

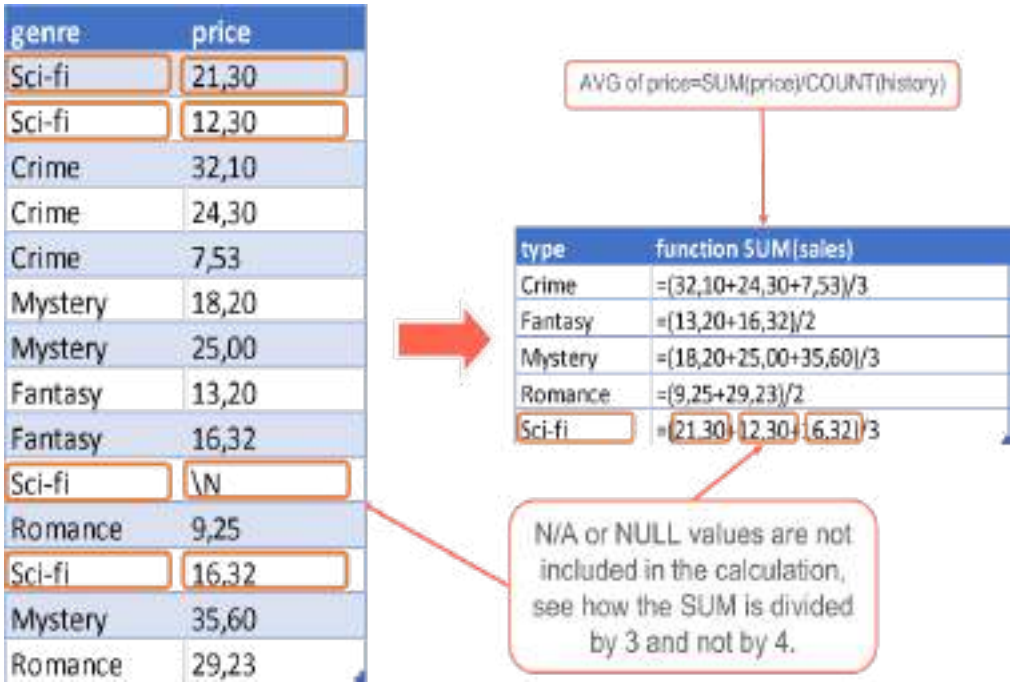
Si queremos conocer el precio promedio por tipo de libro, debemos agregar el campo de precio al área de valores y asegurarnos de utilizar la función promedio.



El resultado es el precio promedio (AVG) por tipo de libro.

Row Labels	Average of price
Crime	21,31
Fantasy	14,76
Mystery	26,26666667
Romance	19,24
Sci-fi	16,64
Grand Total	**20,05**

Operación de agrupamiento:

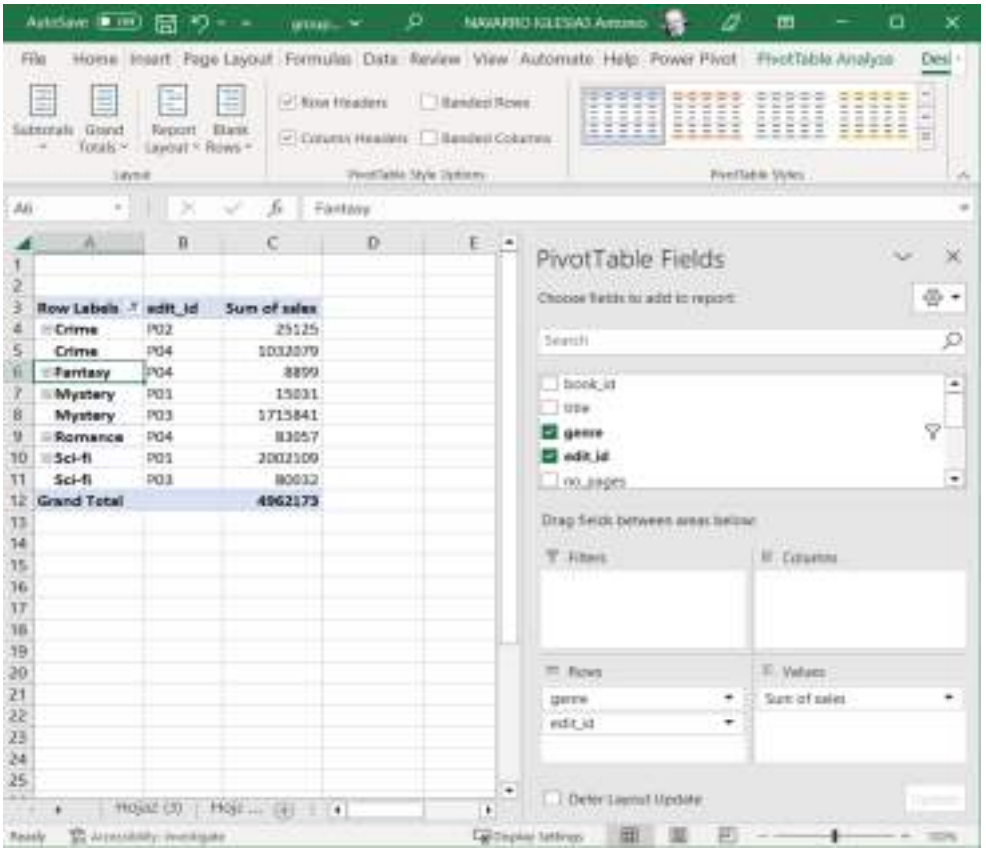


AGRUPANDO POR UNO O MÁS CAMPOS

Como puedes ver, para crear informes de agrupamiento necesitamos dos elementos. El primero es un campo para agrupar y el segundo es una función de agrupamiento (COUNT(), MAX(), MIN(), SUM(), AVG()) aplicada a un campo de la tabla; en los ejemplos anteriores utilizamos 'sales' o 'price' como entrada para la función de agrupamiento.

La agrupación puede realizarse para más de un campo. En la siguiente tabla dinámica incluimos 'pub_id', por lo tanto obtenemos un informe de SUMA de ventas por tipo de publicación ('genre') y referencia de editor ('edit_id').

Obtener esta tabla es un poco más complejo y está fuera del alcance de este capítulo.



La función de agrupamiento en la tabla dinámica funciona con la siguiente lógica.

genre	edit_id	sales
Sci-fi	P01	1520
Sci-fi	P03	80032
Crime	P02	25125
Crime	P04	12054
Crime	P04	1020025
Mystery	P01	15031
Mystery	P03	1689854
Fantasy	P04	5648
Fantasy	P04	3251
Sci-fi	P01	\N
Romance	P04	2536
Sci-fi	P01	2000589
Mystery	P03	25987
Romance	P04	80521

Row Labels	edit_id	function SUM(sales)
Crime	P02	
Crime	P04	
Fantasy	P04	
Mystery	P01	
Mystery	P03	
Romance	P04	
Sci-fi	P01	=1520+2000589
Sci-fi	P03	=80032

Co-funded by the European Union
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

Ahora se calcula la suma de las ventas para aquellos valores que tienen el mismo tipo de publicación y el mismo ID de publicación (pub_id).

GROUP BY en SQL

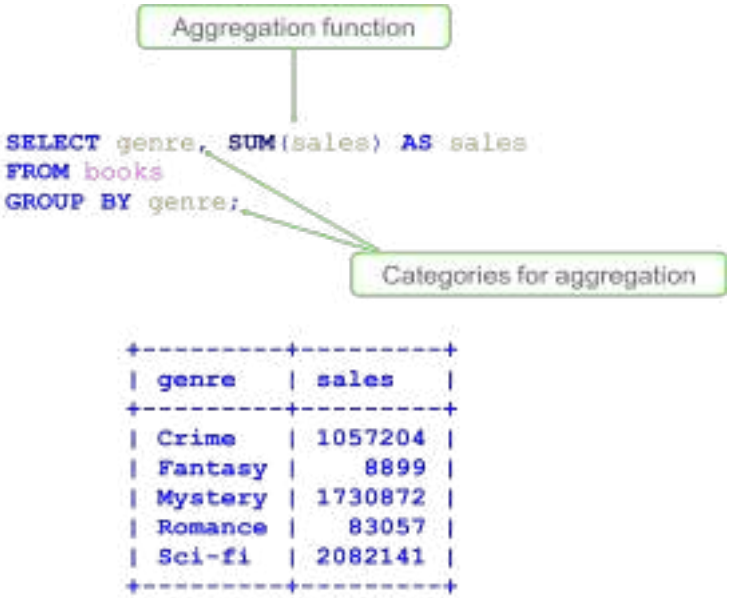
Ahora vamos a reproducir la función de agrupamiento realizada en la tabla dinámica con sentencias SQL sobre nuestra base de datos 'db_books'.

```
SELECT column_1, column_2, agregate_expression(column)

FROM database.table
GROUP BY column_1, column_2, ...
```

El comando GROUP BY:

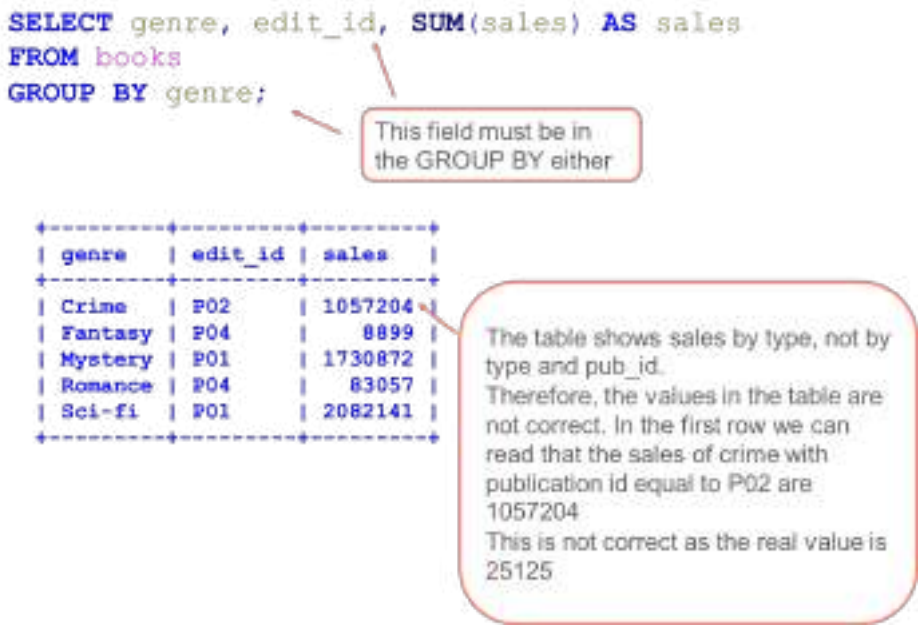
- Divide una tabla en grupos lógicos (categorías) y calcula estadísticas agregadas para cada grupo.
- La cláusula GROUP BY se coloca después de la cláusula WHERE y antes de la cláusula ORDER BY.
- Una columna no puede aparecer en la cláusula SELECT a menos que también esté incluida en la cláusula GROUP BY.
- Usar una cláusula WHERE en una declaración que contiene GROUP BY elimina filas antes de que se formen.

Como hicimos anteriormente en nuestras tablas dinámicas en Excel, necesitamos un informe de **ventas por tipo de publicación.**

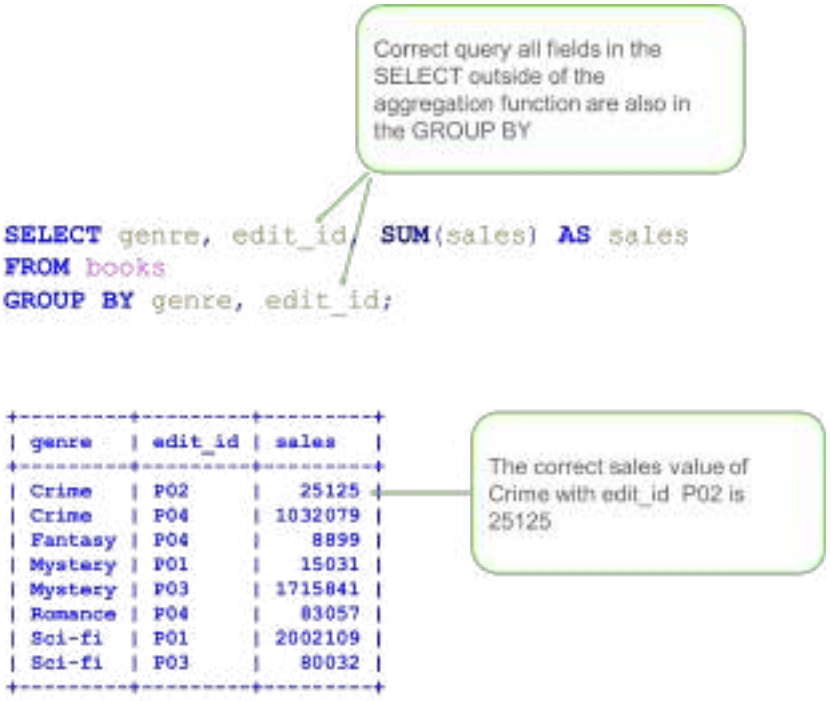


Si agrego otras columnas en el SELECT que no están en la declaración GROUP BY, la consulta no fallará pero el resultado mostrará información incorrecta.

Query incorrecta:

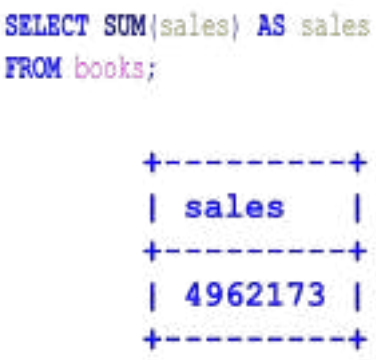


Query Correcto:



AGREGACIÓN SIN GROUP BY

Si eliminamos la palabra clave GROUP BY de la sentencia, obtenemos valores agregados de toda la tabla.



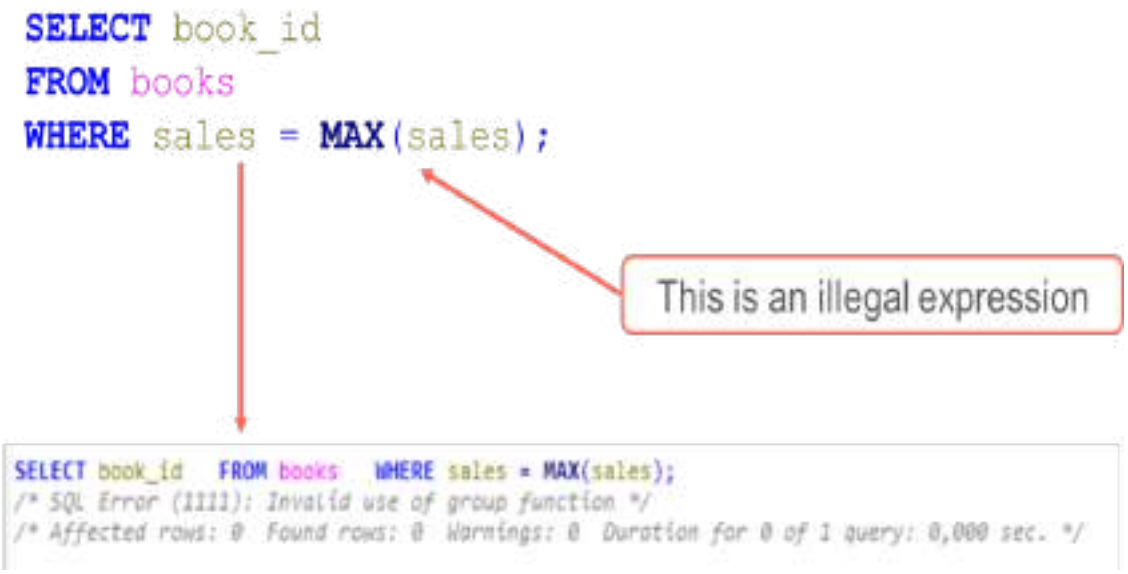
Comentarios importantes:

Algunos comentarios importantes sobre esta función de agregación que son válidos tanto para consultas con, como sin GROUP BY:

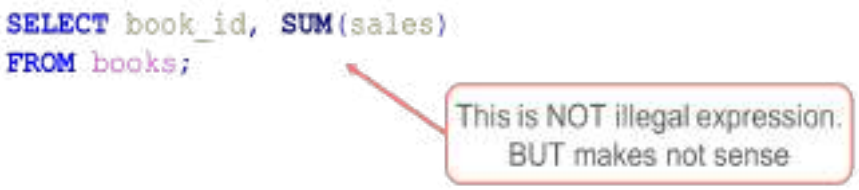
- Las funciones de agregación, excepto COUNT(*), ignoran los valores nulos.

Co-funded by the European Union
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

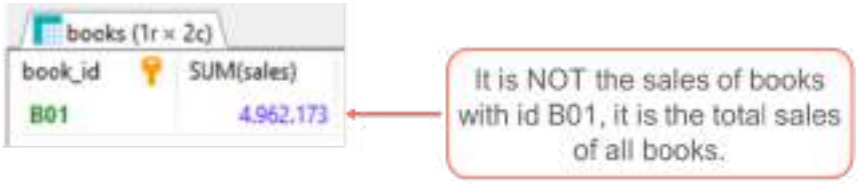
- Las funciones de agregación no pueden aparecer en una cláusula WHERE.



- No se pueden mezclar expresiones agregadas y no agregadas en un SELECT.



El query no falla, pero el resultado es incorrecto.



Puedes verificar fácilmente que B01 tiene un total de ventas de 1,520 unidades.

Co-funded by the European Union
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

- Se pueden utilizar más de una expresión de agregado en una cláusula SELECT.

```
SELECT MIN(sales), MAX(sales)
FROM books;
```

- No se pueden anidar expresiones de agregado.

```
SELECT SUM(AVG(sales))
FROM books;
```

Invalid expression

```
/* SQL Error (1111): Invalid use of group function */
```

- No se pueden usar subqueries en expresiones de agregado.

```
SELECT AVG(SELECT price FROM books);
```

MÁS EJEMPLOS

Aquí veremos algunos ejemplos de consultas de agregación con y sin expresiones de agrupamiento.

"Get maximum price in the table titles"

```
SELECT MAX(price)
FROM books;
```

```
+------------+
| MAX(price) |
+------------+
|      35.60 |
+------------+
```

"Get maximum price for each book category (genre)"

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

```
SELECT SELECT genre, MAX(price)
FROM books;
```

```
+---------+-------------+
| genre   | MAX(price)  |
+---------+-------------+
| Crime   |       32.10 |
| Fantasy |       16.32 |
| Mystery |       35.60 |
| Romance |       29.23 |
| Sci-fi  |       21.30 |
+---------+-------------+
```

"Get the earliest date of publication of a book"

```
SELECT
MIN(pubdate) AS "Earliest pubdate"
FROM books;
```

```
+------------------+
| Earliest pubdate |
+------------------+
| 1992-01-18       |
+------------------+
```

"Get the earliest date of publication of a book for each type of publication"

```
SELECT
    genre
    , MIN(pubdate)
    AS "Earliest pubdate"
FROM books
GROUP BY genre;
```

```
+---------+------------------+
| genre   | Earliest pubdate |
+---------+------------------+
| Crime   | 2000-06-02       |
| Fantasy | 2020-01-17       |
| Mystery | 1995-07-26       |
| Romance | 1992-01-18       |
| Sci-fi  | 1998-05-12       |
+---------+------------------+
```

"Get the lowest book price, the highest book price, and the range of different"

```
SELECT
  MIN(price) AS "lowest price"
, MAX(price) AS "higher price"
, MAX(price) - MIN(price) AS
"difference"
FROM books;
```

```
+--------------+--------------+------------+
| lowest price | higher price | difference |
+--------------+--------------+------------+
|         7.53 |        35.60 |      28.07 |
+--------------+--------------+------------+
```

"Get the lowest and highest price and their differences for each book genre and publisher id with a GROUP BY statement".

```
SELECT
  MIN(price) AS "lowest price"
, MAX(price) AS "higher price"
, MAX(price) - MIN(price) AS "difference"
FROM books
GROUP BY genre, edit_id;
```

```
+---------------+---------------+-------------+
| lowest price | higher price | difference |
+---------------+---------------+-------------+
|         32.10 |         32.10 |        0.00 |
|          7.53 |         24.30 |       16.77 |
|         13.20 |         16.32 |        3.12 |
|         18.20 |         18.20 |        0.00 |
|         25.00 |         35.60 |       10.60 |
|          9.25 |         29.23 |       19.98 |
|         16.32 |         21.30 |        4.98 |
|         12.30 |         12.30 |        0.00 |
+---------------+---------------+-------------+
```

"Get total sales of books published in 2001"

```
SELECT SUM(sales) AS "Sales 01"
FROM books
WHERE pubdate
BETWEEN '2001-01-01'
AND
        '2001-12-31';
```

```
+-----------+
| Sales 01  |
+-----------+
|   2105746 |
+-----------+
```

"Get total sales of books published in 2001 by genre"

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

```
SELECT genre
       , SUM(sales) AS "Sales 01"
       , '2001' AS 'year'
FROM books
WHERE pubdate
BETWEEN '2001-01-01'
    AND
        '2001-12-31'
GROUP BY genre;
```

Notice how I have created a new column with name 'year' and value 2001. The field 'year' does not exist in the books table, I create it to add additional data to the query result table. I am NOT adding new fields to the books table in the database.

```
+--------+----------+------+
| genre  | Sales 01 | year |
+--------+----------+------+
| Crime  |    25125 | 2001 |
| Sci-fi |  2080621 | 2001 |
+--------+----------+------+
```

"I need a small financial report, our accounting manager asked for the total revenue in 2001, he also wants to see the total sales and the average price of the books."

```
SELECT AVG(price) AS 'avg book price'
       , SUM(sales) AS 'total sales'
       , SUM(price*sales) AS 'gross profit'
FROM books
WHERE pubdate
BETWEEN '2001-01-01'
    AND
        '2001-12-31';
```

```
+----------------+-------------+--------------+
| avg book price | total sales | gross profit |
+----------------+-------------+--------------+
|      20.240000 |     2105746 |  34440518.58 |
+----------------+-------------+--------------+
```

Now I want a similar report but for each editorial.

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

```
SELECT edit_id
       , AVG(price) AS 'avg book price'
       , SUM(sales) AS 'total sales'
       , SUM(price*sales) AS 'gross profit'
FROM books
WHERE pubdate
BETWEEN '2001-01-01'
    AND
        '2001-12-31'
GROUP BY edit_id;
```

```
+---------+----------------+-------------+--------------+
| edit_id | avg book price | total sales | gross profit |
+---------+----------------+-------------+--------------+
| P01     |      16.320000 |     2000589 |  32649612.48 |
| P02     |      32.100000 |       25125 |    806512.50 |
| P03     |      12.300000 |       80032 |    984393.60 |
+---------+----------------+-------------+--------------+
```

2.2.11. El HAVING keyword

HAVING + GROUP BY

Recuerda esta limitación en las expresiones de agregación:

"Las funciones de agregación no pueden aparecer en una cláusula WHERE."

```
SELECT edit_id, MAX(sales)
FROM books
WHERE MAX(sales)>1000000
GROUP BY edit_id;
```

```
/* SQL Error
(1111): Invalid use
of group function
*/
```

La razón por la cual esto no es posible es que en el orden de ejecución de una declaración SQL, la cláusula WHERE se ejecuta antes de la palabra clave GROUP BY, por lo que SQL no puede resolver 'WHERE MAX(sales)>1000000', ya que el GROUP BY aún no se ha completado y el motor de la base de datos no puede calcular MAX(sales).

Co-funded by the European Union

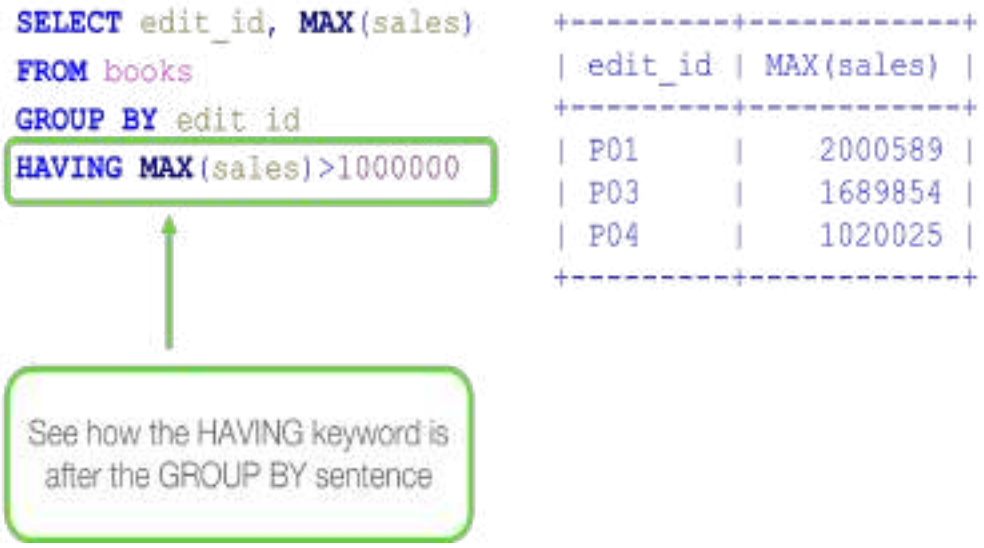
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

Entonces, ¿cómo podemos hacer una condición de filtro sobre valores agregados?
A través de la palabra clave HAVING es posible filtrar en la declaración GROUP BY, de manera similar a como funciona WHERE en la cláusula SELECT.
Orden de ejecución:

- La cláusula **WHERE** filtra las filas resultantes de las operaciones especificadas en **FROM y JOIN.**
- La cláusula **GROUP BY** crea los grupos después de la cláusula WHERE.
- La cláusula **HAVING** filtra las filas del resultado agrupado.

WHERE antes de **GROUP BY**
GROUP BY antes de **HAVING**

EL NOTACION HAVING
La notación correcta para la consulta anterior es:



Co-funded by the European Union
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

ALGUNOS EJEMPLOS MÁS

“Search for states in which more than two writers reside”

```
SELECT state, COUNT(state)
FROM writers
GROUP BY state
HAVING count(state)>1;
```

```
+-------+--------------+
| state | COUNT(state) |
+-------+--------------+
| CA    |            2 |
| NY    |            2 |
+-------+--------------+
```

"Lista de ventas totales y precio promedio por editor con identificadores P02, P03 y P04, incluye en el informe solo aquellos géneros de libros con más de 30,000 ventas totales y un precio promedio inferior a 30."

Aquí hay una consulta con diferentes condiciones. Escribir la declaración podría ser complejo para nuevos aprendices de SQL. En nuestra experiencia, el mejor enfoque para este tipo de problema es dividirlo en diferentes etapas, probar el resultado y luego agregar la condición a la consulta.

Siguiendo esta lógica, veamos cómo podemos obtener el informe.

1. Primero obtengamos los valores agregados totales de toda la tabla.

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

```
SELECT
 SUM(sales) AS 'Sales'
 , AVG(price) AS 'AVG price'
FROM books
```

```
+---------+-----------+
| Sales   | AVG price |
+---------+-----------+
| 4962173 | 20.050000 |
+---------+-----------+
```

2. Ahora, considera solo los valores de libros con edit_id igual a P01, P03 y P04

```
SELECT
    SUM(sales) AS 'Sales'
    , AVG(price) AS 'AVG price'
FROM books
WHERE edit_id IN ('P02', 'P03', 'P04');
                  ;
```

```
+---------+-----------+
| Sales   | AVG price |
+---------+-----------+
| 1149160 | 18.847143 |
+---------+-----------+
```

3. El siguiente paso es agrupar los valores por la categoría 'genre'.

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

```
SELECT genre
       , SUM(sales) AS 'Sales'
       , AVG(price) AS 'AVG price'
FROM books
WHERE edit_id IN ('P02', 'P03', 'P04')
GROUP BY genre;
```

```
+---------+---------+-----------+
| genre   | Sales   | AVG price |
+---------+---------+-----------+
| Crime   | 1057204 | 21.310000 |
| Fantasy |    8899 | 14.760000 |
| Romance |   83057 | 19.240000 |
+---------+---------+-----------+
```

4. Finalmente, es fácil aplicar los filtros en las métricas agregadas utilizando HAVING.

```
SELECT genre
       , SUM(sales)
       , AVG(price)
FROM books
WHERE edit_id IN ('P02', 'P03', 'P04')
GROUP BY genre
HAVING SUM(sales) >30000
    AND
       AVG(PRICE)<20;
```

```
+---------+-------------+-------------+
| genre   | SUM(sales)  | AVG(price)  |
+---------+-------------+-------------+
| Romance |       83057 |   19.240000 |
+---------+-------------+-------------+
```


Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

2.3. Querying más de una tabla

Felicidades! Ahora has completado una buena parte de nuestras lecciones de formación en SQL y estás listo para enfrentar consultas más complejas. En esta próxima lección, aprenderemos cómo escribir consultas que involucren múltiples tablas utilizando las sentencias JOIN y subconsultas. Esta es una habilidad esencial para cualquier programador de SQL, y con este conocimiento podrás crear consultas poderosas y sofisticadas. Al final de esta lección, tendrás la confianza para escribir consultas complejas que desbloquearán el verdadero poder de SQL.

Así que prepárate para llevar tus consultas al siguiente nivel!

2.3.1. El Subqueries

Mejor explicarlo con un ejemplo.

Nos estamos adentrando en temas más complejos cuando hablamos de subconsultas, pero no te preocupes, la lógica detrás de una subconsulta es simple. Una subconsulta devuelve un conjunto de datos, y esta tabla de datos se utiliza como entrada para la consulta principal.

Veámoslo con un ejemplo. **Estamos analizando los datos de nuestros escritores, y por alguna razón queremos saber si algunos de los autores viven en el mismo estado que el nuevo editor en nuestra base de datos.**

Como ya sabemos, el mejor enfoque para este tipo de consultas complejas es dividir el problema en diferentes partes, que montaremos secuencialmente entre sí.

- *Obtener el estado de los nuevos editores es fácil.*

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

```
SELECT state FROM
new_editorial;
```

```
+--------+
| state  |
+--------+
| FL     |
| MA     |
| NULL   |
| NULL   |
+--------+
```

Como resultado de la consulta, obtenemos una tabla con los estados de los nuevos editores: 'FL' y 'MA'.

- *Obtener los autores que viven en Florida (FL) y Massachusetts (MA) también es fácil.*

```
SELECT * FROM writers
WHERE state IN ('FL', 'MA');
```

```
+-------+----------+----------+--------------+------------------+--------+-------+-------+
| wr_id | wr_fname | wr_lname | phone        | address          | city   | state | zip   |
+-------+----------+----------+--------------+------------------+--------+-------+-------+
| W08   | Bruno    | Randolph | 617-824-5229 | 4323 Bates Drive | Boston | MA    | 2110  |
+-------+----------+----------+--------------+------------------+--------+-------+-------+
```

Hecho! Bueno, no del todo. Imaginemos que se añaden nuevos registros en la tabla 'new_publisher', un nuevo editor ubicado en California (CA) es agregado. Eso significa que debo ejecutar la primera consulta, tomar nota de los estados y modificar manualmente la segunda, lo cual no es ideal. ¿Qué pasa si la lista de estados es larga (no solo dos como en el ejemplo)? ¿Qué sucede si cometo un error al escribir los códigos de estado en la declaración IN?

Es evidente que debemos encontrar una manera de ejecutar ambas consultas evitando cualquier configuración manual intermedia.

- *Combinándolo todo*

Existe un método para ejecutar las consultas anteriores combinadas en una sola declaración. El subquery:

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

```
SELECT * FROM writers
WHERE state
IN (SELECT state FROM new_editorial);
```

Embedded sub-query returning a list of the states in 'new editorial' to be used by the IN statement of the main query.

```
+-------+----------+----------+--------------+-----------------+--------+-------+-------+
| wr_id | wr_fname | wr_lname | phone        | address         | city   | state | zip   |
+-------+----------+----------+--------------+-----------------+--------+-------+-------+
| W06   | Bruno    | Randolph | 617-824-5229 | 4323 Bates Drive | Boston | MA    | 2110  |
+-------+----------+----------+--------------+-----------------+--------+-------+-------+
```

ALGUNOS EJEMPLOS MÁS

"Select authors who live in the same state as Emma Tony."

```
SELECT * FROM writers
WHERE state = (
      SELECT state FROM writers
      WHERE wr_fname='Emma'
        AND
            wr_lname='Tony'           );
```

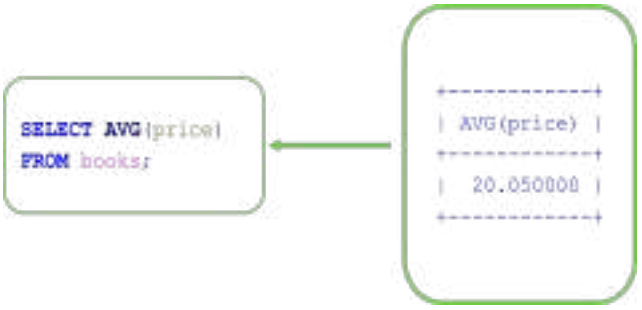
```
+--------+
| state  |
+--------+
| CA     |
+--------+
```

The subquery returns the state in which Emma Tiny lives.

```
+-------+----------+----------+--------------+-------------------+-------------+-------+-------+
| wr_id | wr_fname | wr_lname | phone        | address           | city        | state | zip   |
+-------+----------+----------+--------------+-------------------+-------------+-------+-------+
| W06   |          | Ronuald  | 213-353-4665 | 4623 Brannon Street | Los Angeles | CA    | 90026 |
| W07   | Emma     | Tony     | 714-421-2700 | 1549 Liberty Avenue | Los Angeles | CA    | 90071 |
+-------+----------+----------+--------------+-------------------+-------------+-------+-------+
```

"Selecting books with a price higher than the average price of all books."

El precio promedio de los libros es:



Y los libros con precios superiores al promedio se resuelven integrando la consulta anterior en una cláusula WHERE como una subconsulta.

```
SELECT title, price
FROM books
WHERE price > (
        SELECT AVG(price)
        FROM books
              );
```

```
+-----------------------------------+-------+
| title                             | price |
+-----------------------------------+-------+
| Steel Sanctuary                   | 21.30 |
| Marked for Gold                   | 32.10 |
| Mak of Silence                    | 24.30 |
| Clue of the Painted Turnip        | 25.00 |
| Mystery of the Misshapen Porter   | 35.60 |
| The Devil's Gift                  | 29.23 |
+-----------------------------------+-------+
```

NOTAS SOBRE SUBQUERIES

Como puedes ver en los ejemplos presentados anteriormente, las subconsultas se utilizan ampliamente en cláusulas 'where'.

Pero las subconsultas también pueden estar anidadas en:

- Cláusulas SELECT
- Cláusulas FROM
- Cláusulas WHERE

Un **ejemplo** de subconsulta en la cláusula FROM:

Echa un vistazo a esta sentencia:

```
SELECT titles.title, titles.price
FROM (SELECT * FROM books) AS titles;
```

1. Creamos una tabla a partir de la tabla books obteniendo todos los títulos:

   ```
   SELECT * FROM books
   ```

2. Nombramos la tabla resultante de la subconsulta como titles.
3. Finalmente, usamos esta tabla en la cláusula FROM y seleccionamos las columnas title y price:

```
+----------------------------------+-------+
| title                            | price |
+----------------------------------+-------+
| Steel Sanctuary                  | 21.30 |
| Marked for Gold                  | 32.10 |
| Mak of Silence                   | 24.30 |
| Clue of the Painted Turnip       | 25.00 |
| Mystery of the Misshapen Porter  | 35.60 |
| The Devil's Gift                 | 29.23 |
+----------------------------------+-------+
```

Un ejemplo de subconsulta en cláusulas SELECT

Esto puede no tener mucho sentido, pero imagina un informe como este:

"List the name of the writers and put next to it the total number of writers in our database."

```
SELECT  wr_fname
      , wr_lname
      ,(SELECT COUNT(*) FROM writers) AS countWriters
FROM
      writers;
```

wr_fname	wr_lname	countWwriters
Randolf	Stormy	8
Sam	Christobel	8
Charlene	Osborne	8
Ryker	Fran	8
Alene	Loreen	8
	Romuald	8
Emma	Tony	8
Bruno	Randolph	8

1. El subquery **SELECT COUNT(*) FROM writers** devuelve el número de autores (writers).

COUNT(*)
8

2. Nombramos la tabla resultante de la subconsulta como titles.
3. Finalmente, usamos esta tabla en la cláusula FROM y seleccionamos las columnas title y price:

```
SELECT  wr_fname
      , wr_lname
      ,(SELECT COUNT(*) FROM writers) AS countWriters
FROM
      writers;
```

2.3.2. El JOIN'S

Qué es un JOIN?

Un JOIN es una cláusula utilizada para combinar una o más tablas basadas en una columna común entre ellas.

La declaración SQL para reproducir lo que hicimos anteriormente en Excel es

Bien. No es un concepto fácil. Veámoslo con un ejemplo.

Imagina que tenemos dos tablas:

book_id	title	genre	edit_id
B01	Steel Sanctuary	Sci-fi	P01
B02	The Ruins of Time	Sci-fi	P03
B03	Marked for Gold	Crime	P02
B04	Mak of Silence	Crime	P04
B05	Old Judgment	Crime	P04
B06	Case of the Laughing Goose	Mystery	P01
B07	Clue of the Painted Turnip	Mystery	P03
B08	Druid's Crypt	Fantasy	P04
B09	The Dawn of the Wicked	Fantasy	P04
B10	2619: Gamma	Sci-fi	P01
B11	Heart of Stone	Romance	P04
B12	Hand of Andromeda	Sci-fi	P01
B13	Mystery of the Misshapen Porter	Mystery	P03
B14	The Devil's Gift	Romance	P04

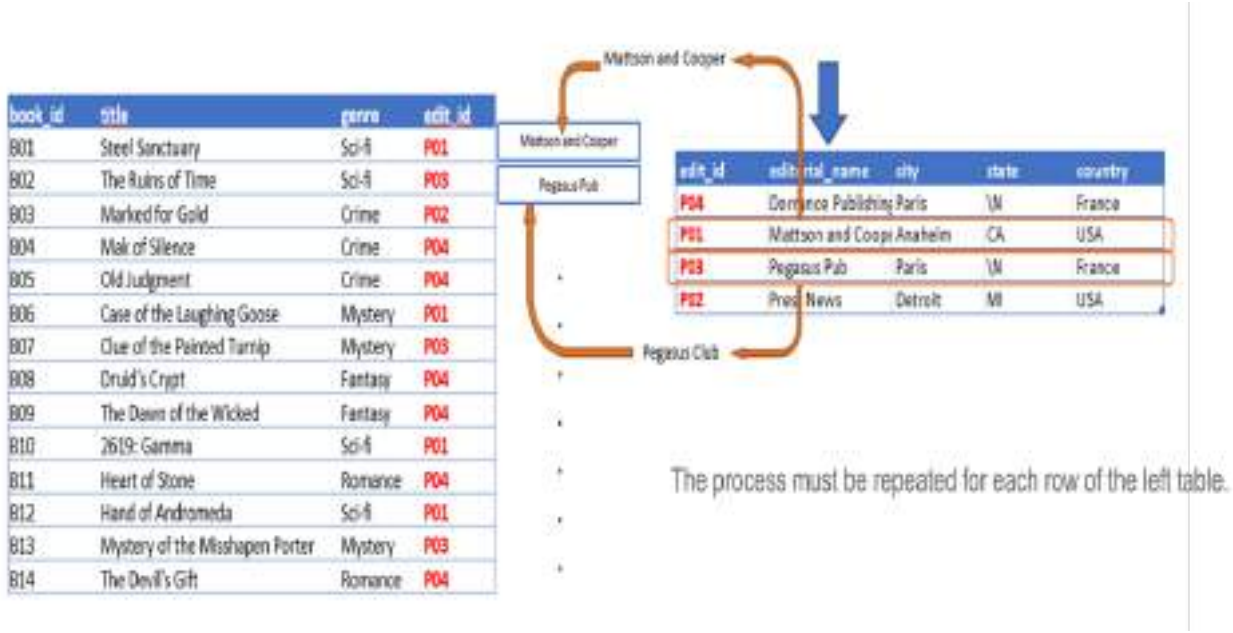
edit_id	editorial_name	city	state	country
P04	Dorrance Publishing	Paris	\N	France
P01	Mattson and Cooper	Anaheim	CA	USA
P03	Pegasus Pub	Paris	\N	France
P02	Press News	Detroit	MI	USA

Como podemos ver en la primera tabla, aparece el código del editor, pero no su nombre ni ninguna otra información sobre él.
Me gustaría agregar esta información en mi tabla de libros, obteniendo una tabla como esta:

book_id	title	genre	edit_id	editorial_name
B01	Steel Sanctuary	Sci-fi	P01	Mattson and Cooper
B02	The Ruins of Time	Sci-fi	P03	Pegasus Pub
B03	Marked for Gold	Crime	P02	Press News
B04	Mak of Silence	Crime	P04	Dorrance Publishing
B05	Old Judgment	Crime	P04	Dorrance Publishing
B06	Case of the Laughing Goose	Mystery	P01	Mattson and Cooper
B07	Clue of the Painted Turnip	Mystery	P03	Pegasus Pub
B08	Druid's Crypt	Fantasy	P04	Dorrance Publishing
B09	The Dawn of the Wicked	Fantasy	P04	Dorrance Publishing
B10	2619: Gamma	Sci-fi	P01	Mattson and Cooper
B11	Heart of Stone	Romance	P04	Dorrance Publishing
B12	Hand of Andromeda	Sci-fi	P01	Mattson and Cooper
B13	Mystery of the Misshapen Porter	Mystery	P03	Pegasus Pub
B14	The Devil's Gift	Romance	P04	Dorrance Publishing

Afortunadamente, tenemos una clave para relacionar ambas tablas, el campo **"edit_id".**
Por lo tanto, debemos tomar nota del valor del campo edit_id en la tabla de libros, por ejemplo, para el título 'Steel Sanctuary', el código del editor ('edit_id') es P01.

Buscamos el editor en la tabla de las editoriales que corresponde a la clave P01, anotamos su nombre y lo añadimos en la tabla de los libros en la fila correspondiente.



book_id	title	genre	edit_id
B01	Steel Sanctuary	Sci-fi	P01
B02	The Ruins of Time	Sci-fi	P03
B03	Marked for Gold	Crime	P02
B04	Mak of Silence	Crime	P04
B05	Old Judgment	Crime	P04
B06	Case of the Laughing Goose	Mystery	P01
B07	Clue of the Painted Turnip	Mystery	P03
B08	Druid's Crypt	Fantasy	P04
B09	The Dawn of the Wicked	Fantasy	P04
B10	2619: Gamma	Sci-fi	P01
B11	Heart of Stone	Romance	P04
B12	Hand of Andromeda	Sci-fi	P01
B13	Mystery of the Misshapen Porter	Mystery	P03
B14	The Devil's Gift	Romance	P04

edit_id	editorial_name	city	state	country
P04	Dominance Publishing	Paris	IM	France
P01	Mattson and Coop	Anaheim	CA	USA
P03	Pegasus Pub	Paris	IM	France
P02	Pres News	Detroit	MI	USA

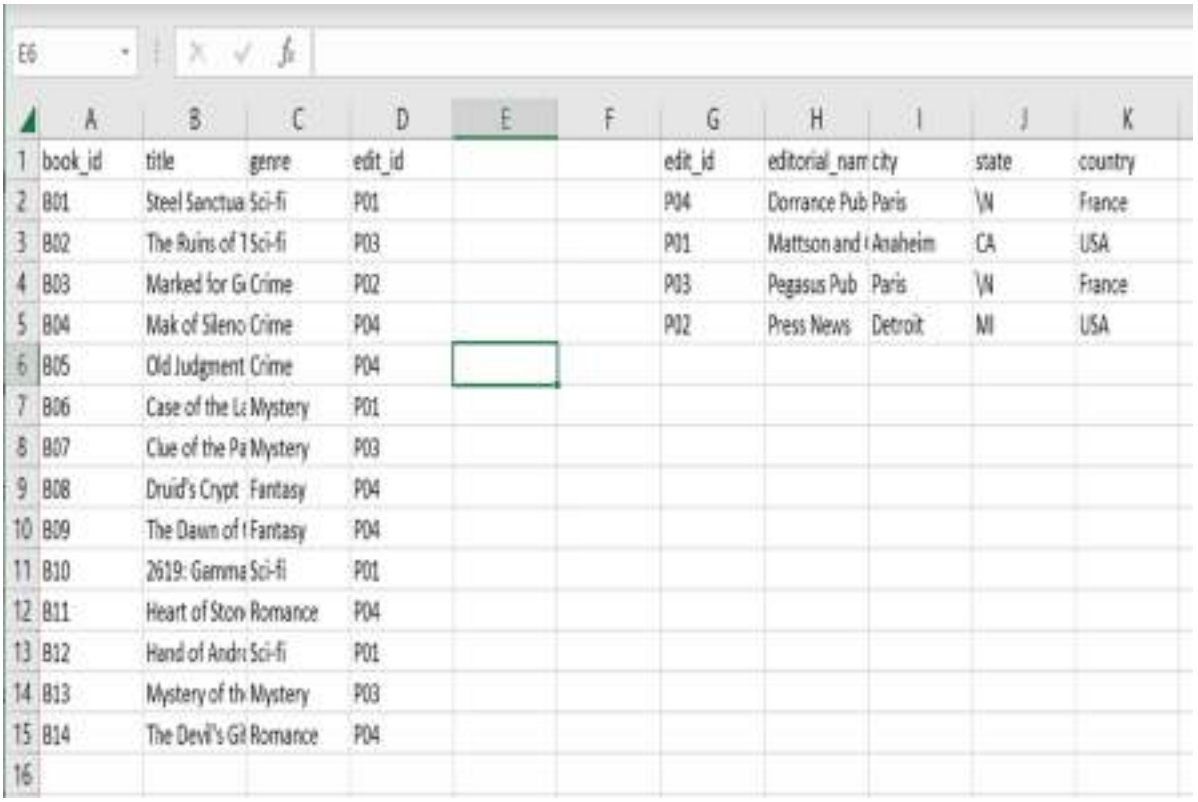
JOINS en Excel, el VLOOKUP

Seguramente sabes cómo hacerlo de manera automática. Al usar la función VLOOKUP, Excel ejecuta automáticamente el proceso explicado anteriormente. Si no estás familiarizado con esta función, veamos cómo usar VLOOKUP en el ejemplo a continuación.

1. Para tu comodidad, hemos copiado/pegado las dos tablas en una única hoja de Excel.

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132



	A	B	C	D	E	F	G	H	I	J	K
1	book_id	title	genre	edit_id			edit_id	editorial_nam	city	state	country
2	B01	Steel Sanctua	Sci-fi	P01			P04	Dorrance Pub	Paris	\N	France
3	B02	The Ruins of T	Sci-fi	P03			P01	Mattson and (	Anaheim	CA	USA
4	B03	Marked for G	Crime	P02			P03	Pegasus Pub	Paris	\N	France
5	B04	Mak of Sileno	Crime	P04			P02	Press News	Detroit	MI	USA
6	B05	Old Judgment	Crime	P04							
7	B06	Case of the L	Mystery	P01							
8	B07	Clue of the Pa	Mystery	P03							
9	B08	Druid's Crypt	Fantasy	P04							
10	B09	The Dawn of t	Fantasy	P04							
11	B10	2619: Gamma	Sci-fi	P01							
12	B11	Heart of Ston	Romance	P04							
13	B12	Hand of Andr	Sci-fi	P01							
14	B13	Mystery of th	Mystery	P03							
15	B14	The Devil's Gl	Romance	P04							
16											

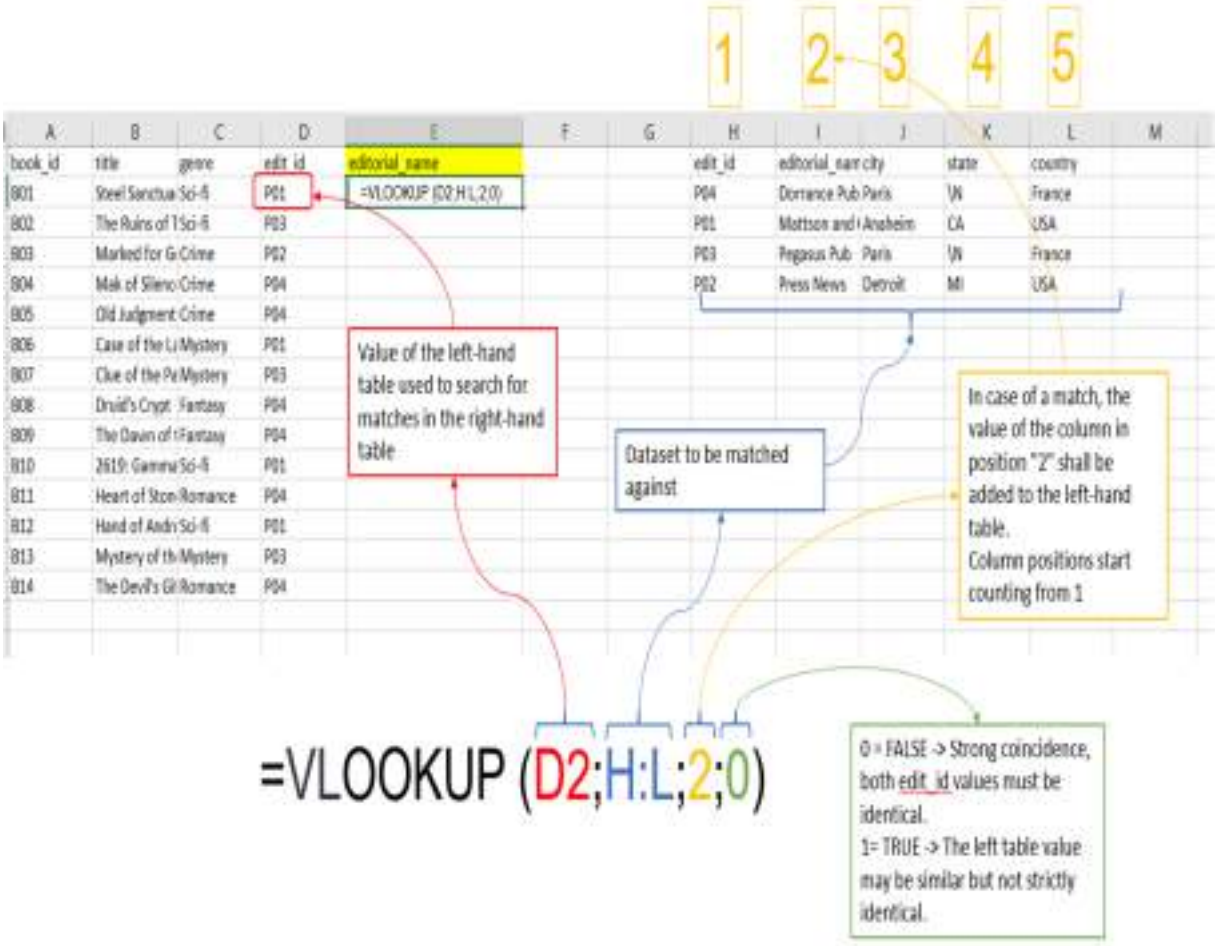
1. Simplemente agrega la columna 'editorial_name' a la tabla izquierda

2. Debajo del encabezado 'editorial_name', agrega la expresión
   =VLOOKUP (D2;H:L;2;0)

3. Arrastra la fórmula al resto de las líneas.

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132



JOINS EN SQL

Entonces, ¡intentemos usar una declaración SQL para reproducir lo que hicimos anteriormente en Excel!

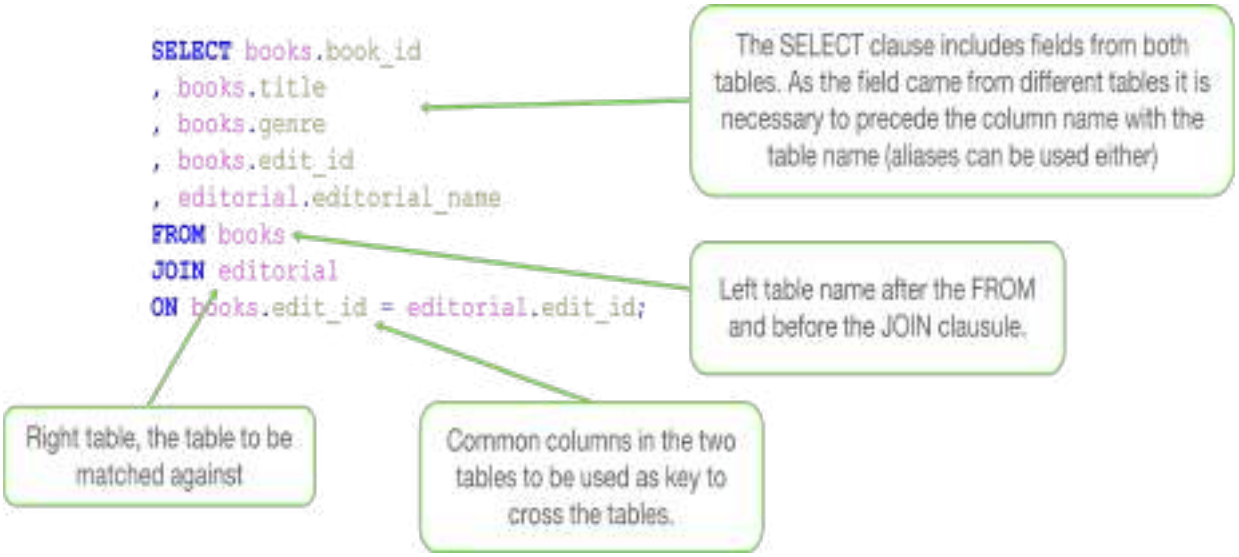
Aunque nos gusta mucho Excel, no es la herramienta ideal para consultar grandes cantidades de datos almacenados en una base de datos. Exportar los valores de una tabla a Excel y luego cruzarlos con **VLOOKUP** puede ser una opción viable para fines educativos o en el caso de tablas pequeñas con pocos registros, pero cuando aumenta la complejidad y el volumen de datos, esta práctica se vuelve poco factible.

Así que intentemos usar una declaración SQL para reproducir lo que hicimos anteriormente en Excel!

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

"Reproduce the result of the above VLOOKUP in your SQL database db_books"



book_id	title	genre	edit_id	editorial_name
B01	Steel Sanctuary	Sci-fi	P01	Mattson and Cooper
B06	Case of the Laughing Goose	Mystery	P01	Mattson and Cooper
B10	2619: Gamma	Sci-fi	P01	Mattson and Cooper
B12	Hand of Andromeda	Sci-fi	P01	Mattson and Cooper
B03	Marked for Cold	Crime	P02	Press News
B02	The Ruins of Time	Sci-fi	P03	Pegasus Pub
B07	Clue of the Painted Turnip	Mystery	P03	Pegasus Pub
B13	Mystery of the Misshapen Porter	Mystery	P03	Pegasus Pub
B04	Mak of Silence	Crime	P04	Dorrance Publishing
B05	Old Judgment	Crime	P04	Dorrance Publishing
B08	Druid's Crypt	Fantasy	P04	Dorrance Publishing
B09	The Dawn of the Wicked	Fantasy	P04	Dorrance Publishing
B11	Heart of Stone	Romance	P04	Dorrance Publishing
B14	The Devil's Gift	Romance	P04	Dorrance Publishing

EL MISMO QUERY USANDO ALIASES PARA LAS TABLES

```
SELECT b.book_id
     , b.title
     , b.genre
     , e.editorial_name
```


Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

```
FROM books AS b
JOIN editorial AS e
ON b.edit_id = e.edit_id
```

TIPOS DE JOIN

Veremos esto con dos tablas de ejemplo. Primero copie y pegue el script SQL en el archivo 'join_tables.sql' en el editor de consultas de Heidi.

Existen diferentes tipos de joins, y la diferencia entre ellos radica en cómo se manejan las filas coincidentes y no coincidentes.
Los JOINs más comunes son:

- **INNER JOIN:** Retorna todas las filas cuando hay al menos una coincidencia en ambas tablas.
- **LEFT JOIN:** Retorna todas las filas de la tabla izquierda y las filas coincidentes de la tabla derecha.
- **RIGHT JOIN:** Retorna todas las filas de la tabla derecha y las filas coincidentes de la tabla izquierda.
- **OUTER JOIN:** Retorna todas las filas de ambas tablas, también conocido como **FULL OUTER JOIN.**

Veremos esto con dos tablas de ejemplo. Primero, copia y pega el script SQL en el archivo 'join_tables.sql' en el editor de consultas de Heidi.

Ejemplo:

Esto creará la base de datos 'join_tables' con las tablas 'sales' y 'country'. Ambas tablas pueden estar vinculadas a través de los campos 'country_id' en la tabla 'sales' y 'count_id' en la tabla 'country'.

Ten en cuenta que el campo que vincula las tablas no tiene que tener el mismo nombre en ambas tablas.

Un rápido análisis de los datos en la base de datos 'join_tables' muestra que los identificadores C01, C02 y C03 están presentes en "sales" y "country", mientras que los valores C04 y C05 no son comunes en ambas tablas.

sales

sale_id	date	country_id	sales
S01	01/01/2020	C01	205
S02	02/01/2020	C03	150
S03	03/02/2020	C01	1125
S04	04/02/2020	C02	721
S05	05/03/2020	C04	320

country

count_id	country_name
C01	USA
C02	Canada
C03	France
C05	Spain

INNER JOIN

Un INNER JOIN recuperará solo las filas donde sales.country_id y country.count_id sean iguales, por lo tanto, no veremos en el resultado de

la consulta ningún valor para las filas donde están presentes C04 o C05, ya que no coinciden en ambas tablas.
Queremos agregar el nombre del país a la tabla sales.

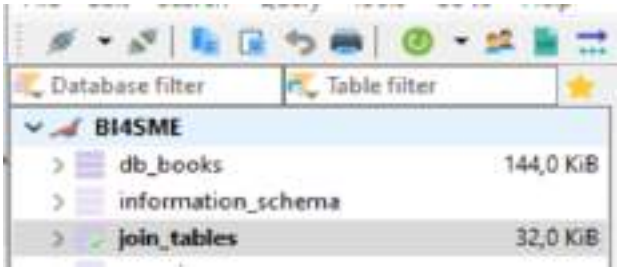
```
SELECT s.sale_id, s.date, s.country_id
      , c.country_name, s.sales
FROM sales s
INNER JOIN country c
ON s.country_id=c.count_id;
```

See how you can refer to the INNER JOIN simply as JOIN.

```
SELECT s.sale_id, s.date, s.country_id
      , c.country_name, s.sales
FROM sales s
JOIN country c
ON s.country_id=c.count_id;
```

Recuerda cambiar a la base de datos "join_tables".

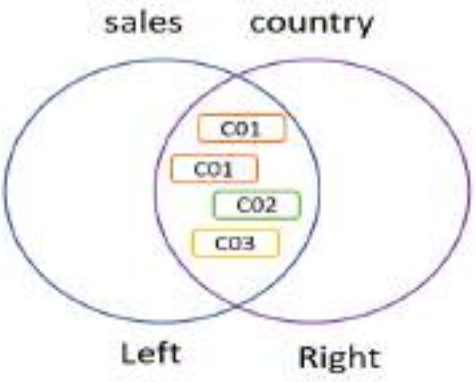
- *En "HEIDI"*



- *Eb el comando prompt*

```
MariaDB [join_tables]> use join_tables
Database changed
MariaDB [db_books]>
```


Co-funded by the European Union
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132



sale_id	date	country_id	country_name	sales
S01	2020-01-01	C01	USA	205
S02	2020-01-02	C03	France	150
S03	2020-02-03	C01	USA	1125
S04	2020-02-04	C02	Canada	721

LEFT JOIN

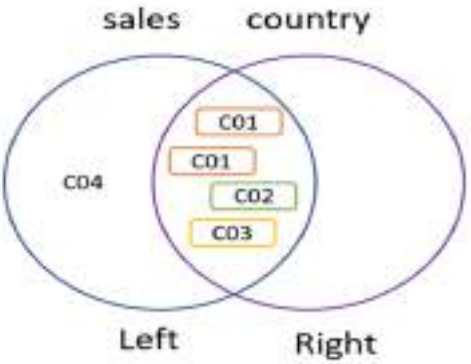
La tabla izquierda es 'sales'. En un LEFT JOIN, todas las filas de las columnas de la izquierda están presentes en el resultado de la consulta, por lo tanto, la fila que contiene los valores C04 será incluida. (Funciona exactamente como la función VLOOKUP en Excel).

La misma consulta que arriba pero todas las filas de la tabla 'sales' deben mostrarse.

```
ELECT s.sale_id, s.date, s.country_id
     , c.country_name, s.sales
FROM sales s
LEFT JOIN country c
ON s.country_id=c.count_id;
```


Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132



sale_id	date	country_id	country_name	sales
S01	2020-01-01	C01	USA	205
S02	2020-01-02	C03	France	150
S03	2020-02-03	C01	USA	1125
S04	2020-02-04	C02	Canada	721
S05	2020-03-05	C04	NULL	320

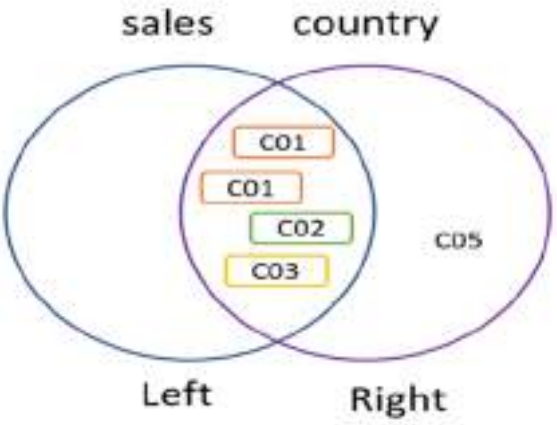
There is no match for country_id=C4 in the table 'country', so the query returns NULL.

RIGHT JOIN

La tabla correcta es 'countries'. Un RIGHT JOIN incluirá todas las filas coincidentes y los datos no coincidentes de la tabla correcta, por lo que se incluirá la fila C05, incluso si no hay un campo coincidente en sales.

Repite la consulta, pero ahora deben mostrarse todas las filas de la tabla 'countries'.

```
SELECT s.sale_id, s.date, s.country_id
      , c.country_name, s.sales
FROM sales s
RIGHT JOIN country c
ON s.country_id=c.count_id;
```



sale_id	date	country_id	country_name	sales
S01	2020-01-01	C01	USA	205
S02	2020-01-02	C03	France	150
S03	2020-02-03	C01	USA	1125
S04	2020-02-04	C02	Canada	721
NULL	NULL	NULL	Spain	NULL

No hay ninguna fila en "sales" con country_id=C05, por lo tanto, todos los campos en la tabla izquierda son NULL en esta fila, pero hay un count_id=C05 en la tabla "country" correspondiente a "Spain". Como se trata de un LEFT JOIN, 'Spain' se incluye en el resultado.

OUTER JOIN

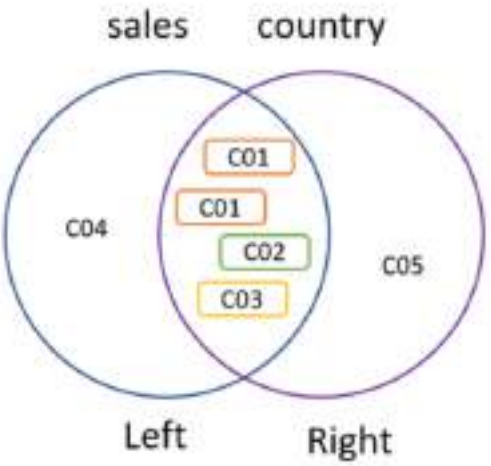
Query en standard SQL.

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

Standard Sintaxis

```
SELECT s.sale_id
      , s.date
      , s.country_id
      , c.country_name
      , s.sales
FROM sales s
OUTER JOIN country c
ON s.country_id=c.count_id;
```



Desafortunadamente, la palabra clave OUTER JOIN no funciona en las bases de datos MariaDB y MySQL; si intentas la consulta anterior en Heidi, fallará.

Podemos ejecutar un OUTER JOIN en MariaDB como una unión de un LEFT JOIN y un RIGHT JOIN utilizando la palabra clave **UNION.**

```
SELECT s.sale_id, s.date, s.country_id
     , c.country_name, s.sales
FROM sales s
LEFT JOIN country c
ON s.country_id=c.count_id

UNION

SELECT s.sale_id, s.date, s.country_id
     , c.country_name, s.sales
FROM sales s
RIGHT JOIN country c
ON s.country_id=c.count_id;
```

sale_id	date	country_id	country_name	sales
S01	2020-01-01	C01	USA	205
S02	2020-01-02	C03	France	150
S03	2020-02-03	C01	USA	1125
S04	2020-02-04	C02	Canada	721
S05	2020-03-05	C04	NULL	320
NULL	NULL	NULL	Spain	NULL

Co-funded by the European Union

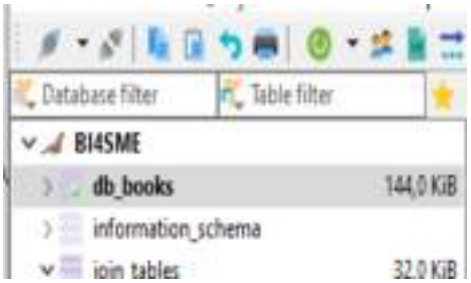
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

JOIN Ejemplos

"Get a report of writers living in state where there is a publishing house."

Recuerda cambiar a la base de datos "db_books".

- *En "HEIDI"*



- *N el comando PROMT*

MariaDB [join_tables]> use db_books
Database changed
MariaDB [db_books]>

```
SELECT w.wr_fname
     , w.wr_lname
     , e.state
     , e.editorial_name

FROM writers w
JOIN editorial e
ON w.state = e.state;
```

```
+----------+----------+-------+--------------------+
| wr_fname | wr_lname | state | editorial_name     |
+----------+----------+-------+--------------------+
| Randolf  | Stormy   | MI    | Press News         |
|          | Romuald  | CA    | Mattson and Cooper |
| Emma     | Tony     | CA    | Mattson and Cooper |
+----------+----------+-------+--------------------+
```

El mismo query como la anterior pero añadiendo ***"authors living or not in the same state of a publisher".***

```
SELECT w.wr_fname
     , w.wr_lname
     , e.state
     , e.editorial_name

FROM writers w
LEFT JOIN editorial e
ON w.state = e.state;
```

wr_fname	wr_lname	state	editorial_name
	Romuald	CA	Mattson and Cooper
Emma	Tony	CA	Mattson and Cooper
Randolf	Stormy	MI	Press News
Sam	Christobel	NULL	NULL
Charlene	Osborne	NULL	NULL
Ryker	Fran	NULL	NULL
Alene	Loreen	NULL	NULL
Bruno	Randolph	NULL	NULL

"List the authors and their books."

Las tablas books y writers no tienen un campo común para cruzarlas, pero hay una tercera tabla llamada "books_writers" que correlaciona book_id con writers_id.

Podemos resolver la consulta uniendo tres tablas, utilizando la tabla books_writers como la tabla principal o aquella con la mayor cardinalidad. Luego cruzamos 'books' y 'writers' con 'books_writers'.

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

```
SELECT w.wr_fname
      , w.wr_lname
      , b.title
FROM books_writers bw
JOIN writers w
ON bw.wr_id = w.wr_id
JOIN books b
ON bw.book_id=b.book_id;
```

1.- Cross 'books_writters' with 'writers'.

2.- Cross 'books_writters' with 'books'.

wr_fname	wr_lname	title
Randolf	Stormy	Steel Sanctuary
Randolf	Stormy	The Ruins of Time
Randolf	Stormy	Mystery of the Misshapen Porter
Sam	Christobel	Case of the Laughing Goose
Sam	Christobel	Clue of the Painted Turnip
Sam	Christobel	2619: Gamma
Sam	Christobel	Hand of Andromeda
Charlene	Osborne	Mak of Silence
Charlene	Osborne	Heart of Stone
Ryker	Fran	Mak of Silence
Ryker	Fran	Old Judgment
Ryker	Fran	Clue of the Painted Turnip
Ryker	Fran	Heart of Stone
Alene	Loreen	Marked for Gold
	Romuald	Druid's Crypt
	Romuald	The Dawn of the Wicked
	Romuald	Heart of Stone
Bruno	Randolph	The Devil's Gift

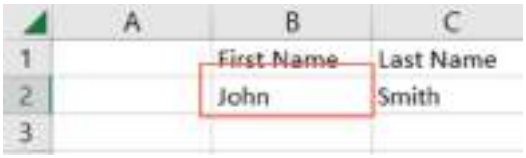
2.4. Trabajando con cadenas de texto

Las funciones de SQL utilizadas para manipular cadenas de texto (**string functions**), comúnmente llamadas funciones de cadena, son una de las herramientas más importantes de SQL.

Pero primero, qué significa "texto"?

Un **texto o cadena de texto**, también conocida simplemente como string en inglés, es un grupo de caracteres que se utilizan como datos en un programa de hoja de cálculo. Las cadenas de texto están compuestas más **comúnmente por palabras, pero también pueden incluir letras, números, caracteres especiales, el símbolo de guion o el numeral.**

Ejemplo → cada celda con datos corresponde a una cadena de texto:



En esta subsección, vamos a ver las 3 formas más comunes e importantes en las que puedes realizar operaciones en cadenas de texto.

2.4.1. El CONCAT keyword

Definición de Concatenación:

La concatenación no agrega espacios entre cadenas de caracteres. por ejemplo 'a' || 'hacer' produce 'todo'.

La concatenación no agrega espacios entre cadenas de caracteres.
Por ejemplo 'a' || 'hacer' produce 'todo'.

- Concatenación es el proceso de combinar el contenido de dos o más columnas en una sola columna.

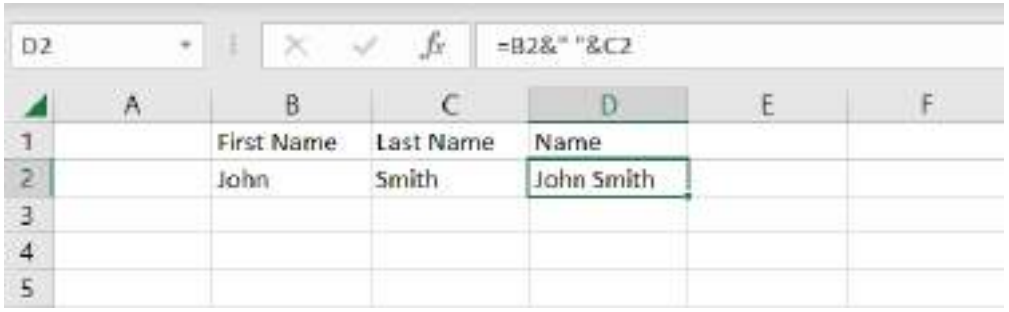
- Por ejemplo, tengo el nombre 'John' en la columna_1 y **'Smith'** en la columna_2, y necesito crear una nueva columna_3 con el nombre completo **'John Smith'.**

Algunos puntos clave:

- La concatenación funciona solo con columnas de tipo cadena de texto.
- La concatenación no añade espacios entre las cadenas de caracteres. Por ejemplo, 'to' || 'do' produce 'todo'.
- El resultado de una concatenación que incluye un valor nulo es nulo.

Excel case

En Excel, la concatenación se realiza con el operador '&



IMPORTANTE: La concatenación no añade espacios. Debes concatenar un espacio en blanco explícitamente si deseas incluirlo.

=B2&" "&C2

De lo contrario, obtendrás 'JohnSmith' en lugar de 'John Smith'

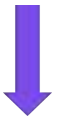
Concatenar en MariaDB

Ejemplo 1.

Concatenación de las columnas del nombre y apellido de los escritores en una nueva columna llamada 'name'.

Co-funded by the European Union
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

```
SELECT wr_fname
       , wr_lname
       , CONCAT(wr_fname, " ", wr_lname) AS 'name'

FROM writers ;
```



wr_fname	wr_lname	name
Randolf	Stormy	Randolf Stormy
Sam	Christobel	Sam Christobel
Charlene	Osborne	Charlene Osborne
Ryker	Fran	Ryker Fran
Alene	Loreen	Alene Loreen
	Romuald	Romuald
Emma	Tony	Emma Tony
Bruno	Randolph	Bruno Randolph

Aunque esta es una **función de cadena** de texto, **puede manejar argumentos numéricos** (y cadenas binarias). Cualquier valor numérico se **convierte en su equivalente cadena binaria.**

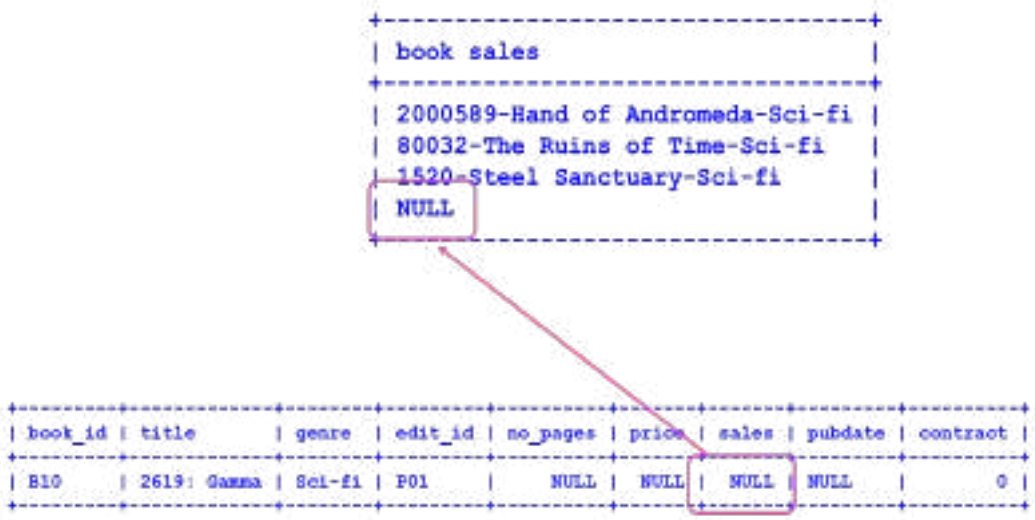
Ejemplo 2.

Lista los libros de ciencia ficción más vendidos e incluye toda la información en una sola columna.

```
SELECT CONCAT (sales,"-", title, "-",genre)
AS 'book sales'
FROM books
WHERE genre='Sci-fi'
ORDER BY sales DESC;
```

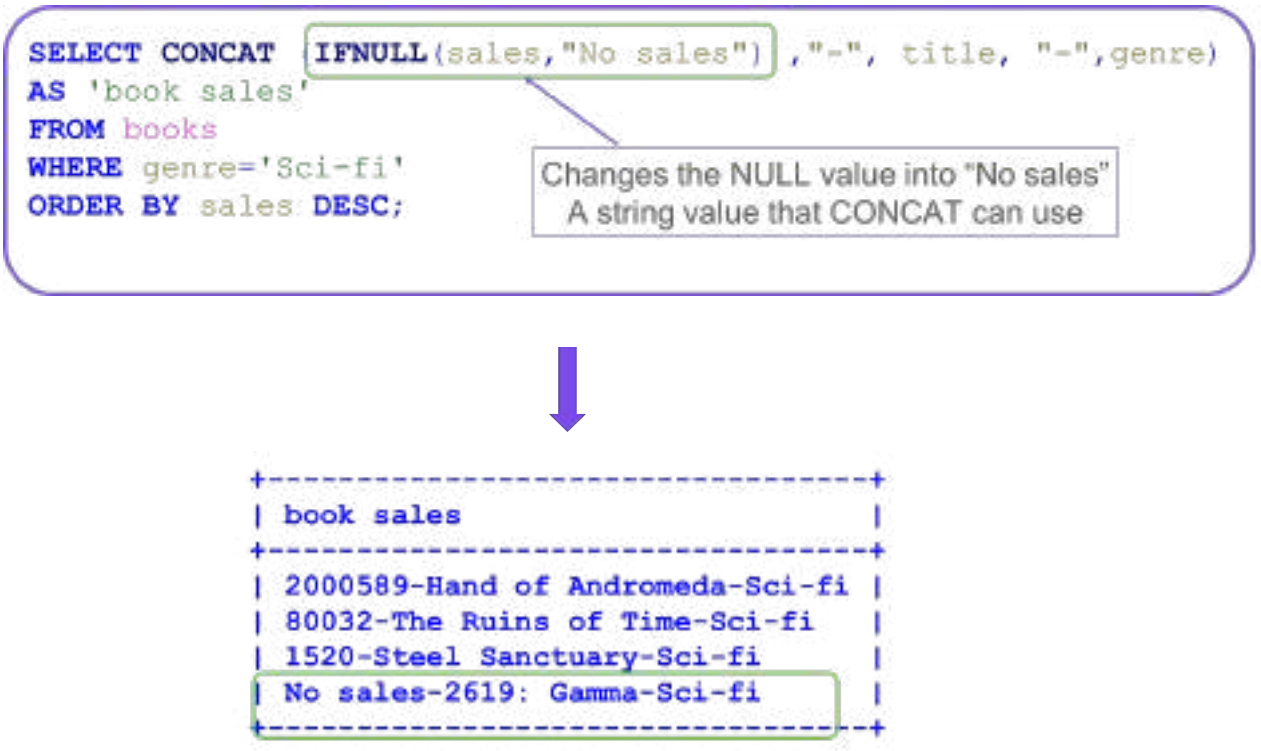
Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132



- Observa cómo SQL maneja los valores NULL en las ventas para la fila con book_id 'B10'.
- MariaDB intenta convertir un valor NULL a una cadena, el motor de SQL no puede hacerlo y devuelve NULL para esta fila.

Una solución con una conversión de NULL a valor de cadena:



Co-funded by the European Union
GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

2.4.2. Substring

Substring significa extraer **una parte de una cadena de texto.**
IMPORTANTE: La notación de substring en MariaDB difiere del SQL estándar.

- Standard SQL

SUBSTRING(string FROM start [FOR length])

- MariaDB

SUBSTR(string, start [,length])

Ejample 1:

Tengo la cadena "221 Baker Street" y quiero dividir la dirección en dos columnas: columna_1 con el número y columna_2 con la calle.

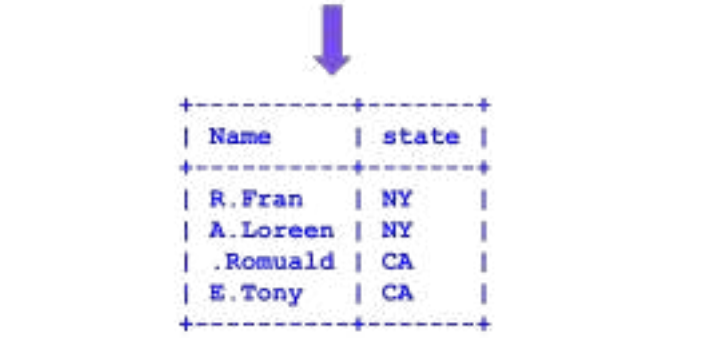

```
SELECT SUBSTR("221 Baker Street", 1 , 3) AS 'number'
     , SUBSTR("221 Baker Street", 5 , 17) AS 'Street';
```

```
+--------+---------------+
| number | Street        |
+--------+---------------+
| 221    | Baker Street  |
+--------+---------------+
```

Ejample 2:

Quiero tener una lista de los nombres de los autores que viven en NY o CA con este formato "R.Stormy", "S.Christobel"....

```
SELECT CONCAT(SUBSTR(wr_fname, 1 , 1)
  , "."
  , wr_lname) AS Name
  , state
FROM writers
WHERE state IN ('NY', 'CA')
```



```
+----------+-------+
| Name     | state |
+----------+-------+
| R.Fran   | NY    |
| A.Loreen | NY    |
| .Romuald | CA    |
| E.Tony   | CA    |
+----------+-------+
```

Ejample 3:

“Lista de autores cuyo número de teléfono comienza con 212"

```
SELECT wr_fname, wr_lname, phone
FROM writers
WHERE SUBSTR(phone,1,3)='212';
```

```
+----------+----------+--------------+
| wr_fname | wr_lname | phone        |
+----------+----------+--------------+
| Ryker    | Fran     | 212-771-4680 |
| Alene    | Loreen   | 212-834-6469 |
+----------+----------+--------------+
```


Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

2.4.3. El LOWER & UPPER keywords

- LOWER devuelve una cadena en la que las mayúsculas se convierten en minúsculas

LOWER(string)

- UPPER devuelve una cadena en la que las minúsculas se convierten en mayúsculas.

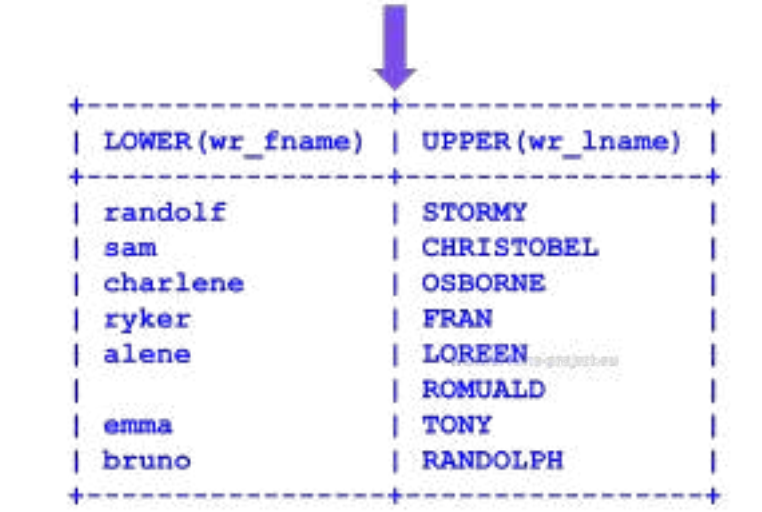
UPPER(string)

- En ambos casos, los dígitos, los caracteres de puntuación y los espacios en blanco se mantienen como están.

Ejemplo 1:

Para cada autor, imprimir el nombre en minúsculas y el apellido en mayúsculas.

```
SELECT LOWER(wr_fname)
     , UPPER(wr_lname)
FROM writers;
```



LOWER(wr_fname)	UPPER(wr_lname)
randolf	STORMY
sam	CHRISTOBEL
charlene	OSBORNE
ryker	FRAN
alene	LOREEN
	ROMUALD
emma	TONY
bruno	RANDOLPH

2.5. Contenido adicional

2.5.1. Trabajando con datos

Trabajar con fechas siempre es complicado, ya sea que estés programando en Python o escribiendo una sentencia SQL. Algunos problemas típicos incluyen:

- A veces, la columna de fecha no es reconocida por el programa como un formato de fecha simplemente porque el tipo de las columnas es cadena y no fecha.
- Otras veces solo queremos tener un elemento de la fecha, por ejemplo, el año.
- Es muy común tener un formato de fecha (por ejemplo, '01-02-2022') y necesitarlo en formato internacional '2022-02-01'.

DÍA, MES, AÑO: CÓMO OBTENERLO

MariaDB no utiliza la notación estándar de SQL para extraer elementos de una fecha.

SQL Standard	MariaDB
EXTRACT([DAY|MONTH|YEAR] FROM pubdate	DAY(pubdate) MONTH(pubdate) YEAR(pubdate)

"List the publication year of books."

```
SELECT DISTINCT YEAR(pubdate) AS pub_year
FROM books
ORDER BY pub_year DESC;
```

```
+----------+
| pub_year |
+----------+
|     2020 |
|     2015 |
|     2014 |
|     2010 |
|     2001 |
|     2000 |
|     1998 |
|     1995 |
|     1992 |
|     NULL |
+----------+
```

"Print books that have been published in the first 6 months of the years 2001 or 2002."

```
SELECT title, pubdate
FROM books
WHERE YEAR(pubdate) BETWEEN 2000 AND 2010
AND
MONTH(pubdate) BETWEEN 1 AND 6 ;
```

```
+------------------+------------+
| title            | pubdate    |
+------------------+------------+
| The Ruins of Time | 2001-04-15 |
| Marked for Gold  | 2001-05-20 |
| Mak of Silence   | 2010-06-12 |
| Old Judgment     | 2000-06-02 |
+------------------+------------+
```

Obtener los valores de la FECHA DE HOY.

'Hoy' o 'fecha actual' es el momento en que se ejecuta la consulta.

Puede ser útil agregar el día actual en el momento de la ejecución de la consulta.

- El motor de búsqueda de SQL obtiene el día actual del reloj del sistema operativo del servidor SQL (en tu caso, del reloj de tu PC).

Fecha CURRENT_DATE

Hora CURRENT_TIME

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

Fecha y Hora CURRENT_TIMESTAMP

Inténtalo así:

```
SELECT
   CURRENT_DATE AS 'DATE'
  ,CURRENT_TIME AS 'TIME'
  ,CURRENT_TIMESTAMP 'DATE & TIME';
```

```
+------------+----------+---------------------+
| DATE       | TIME     | DATE & TIME         |
+------------+----------+---------------------+
| 2023-01-20 | 13:11:42 | 2023-01-20 13:11:42 |
+------------+----------+---------------------+
```

"¿Qué libros se publicaron en el mismo mes en que estoy?"

El resultado de esta consulta variará obviamente según el mes del año en el que se ejecute el script. Yo estoy en enero al momento de escribir esto; cuando ejecutes esta consulta probablemente estarás en otro mes, por lo tanto, tu resultado será diferente.


```
+---------------+--------------+------------------+
| title         | today's MONTH | publication_date |
+---------------+--------------+------------------+
| Druid's Crypt |            1 | 2020-01-17       |
| Heart of Stone |            1 | 1992-01-18       |
+---------------+--------------+------------------+
```

Cambiar el formato de la fecha

El cambio de formato de fecha convierte el formato de la fecha, por ejemplo, de '2023-01-20' a '20,01,2023'.

```
SELECT DATE_FORMAT('2023-01-20', '%d.%m.%X');
```

Otros ejemplos:

```
SELECT DATE_FORMAT('2009-10-04 22:23:00', '%W %M %Y');
SELECT DATE_FORMAT('2007-10-04 22:23:00', '%H:%i:%s');
SELECT DATE_FORMAT('2007-10-04 22:23:00', '%d-%m-%y');
SELECT DATE_FORMAT('2007-10-04 22:23:00', '%d-%m-%Y');
SELECT DATE_FORMAT('1900-10-04 22:23:00', '%D %y %a %d %m %b %j');
SELECT DATE_FORMAT('1997-10-04 22:23:00', '%H %k %I %r %T %S %w');
SELECT DATE_FORMAT('1999-01-03', '%X %V');
SELECT DATE_FORMAT('2006-06-00', '%d');
```

Patrones de fecha más comunes:

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

Option	Description
%d	Day of the month, numeric (00..31)
%m	Month, numeric (00..12)
%Y	Year, numeric, four digits
%y	Year, numeric (two digits)
%H	Hour (00..23)
%i	Minutes, numeric (00..59)
%s	Seconds (00..59)
%a	Abbreviated weekday name (Sun..Sat)
%b	Abbreviated month name (Jan..Dec)
%W	Full weekday name (Sunday..Saturday)
%M	Full month name (January..December)

2.5.2. La particularidad de COUNT

EL USO DE LA PALABRA CLAVE COUNT

El comportamiento de COUNT con los valores nulos difiere según la expresión utilizada.

- **COUNT(expression)** devuelve el número de filas en las columnas; si hay valores NULL, estas filas **NO se cuentan.**
- **COUNT(*)** devuelve el número de filas de una tabla, incluidos los valores NULL; los valores NULL **se cuentan.**

Veámoslo en un ejemplo.

Just get the number of rows in the table books.

```
SELECT COUNT(*)
FROM books;
+----------+
| COUNT(*) |
+----------+
|       14 |
+----------+
```

If I just count by book_id the result must be identical.

```
SELECT COUNT(book_id)
FROM books;
+----------------+
| COUNT(book_id) |
+----------------+
|             14 |
+----------------+
```

Veamos qué pasa si usamos la columna 'pubdate' en lugar de book_id.

```
SELECT COUNT(pubdate)
FROM books;
```

```
+----------------+
| COUNT(pubdate) |
+----------------+
|             13 |
+----------------+
```

In this case the result is 13, the reason is that in the column 'pubdate' we have 13 different values and one NULL, the null value is not accounted.

2.5.3. El DISTINCT en agregaciones

La palabra clave DISTINCT se puede usar para aplicar las funciones de agregación a valores distintos, de modo que los valores duplicados solo se cuenten una vez.

Una aplicación típica es contar valores únicos, en la columna 'edit_id' en libros hay varios valores duplicados. Si queremos saber cuántos editores diferentes tenemos asociados con nuestros libros, la consulta a realizar es:

```
SELECT COUNT(DISTINCT edit_id)
FROM books;
```

```
+--------------------------+
| COUNT(DISTINCT edit_id) |
+--------------------------+
|                        4 |
+--------------------------+
```

Count of different rows, excluding duplicates

See the difference

```
SELECT COUNT(edit_id)
FROM books;
```

```
+-----------------+
| COUNT(edit_id) |
+-----------------+
|              14 |
+-----------------+
```

Row count, including duplicates

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

DISTINCT se puede usar dentro de la expresión de otras funciones de agregación como **SUM o AVG.**

MAS EJEMPLOS

Podemos usar la tabla "Royalties" para hacer algunos ejemplos.

Ya sabemos cómo crear una consulta para verificar si tenemos valores duplicados en el campo 'royalty_rate' en la tabla royalties. ¡Así que comencemos primero con esto!

"Checking for duplicate values in royalty fees"

```
SELECT royalty_rate
      , COUNT(royalty_rate) AS
count
FROM royalties
GROUP BY royalty_rate;
```

Herein we can see duplicated values for the royalty_rates in 0.02, 0.03, and 0.08.

royalty_rate	count
NULL	0
0.01	1
0.02	2
0.03	3
0.05	1
0.06	1
0.08	2
0.09	1
0.10	1
0.11	1

Puede que no tenga ninguna lógica comercial o financiera, pero calculemos la suma de la tasa de regalías, la misma métrica pero sin considerar valores duplicados, el promedio de la tasa de regalías y la misma relación sin valores duplicados,

Comparando ratios con y sin valores duplicados

```
SELECT
    SUM(royalty_rate) AS "rates sum"
   ,SUM(DISTINCT royalty_rate)  AS "sum unique rates"
   ,avg(royalty_rate) AS "rates avg"
   ,avg(DISTINCT royalty_rate) AS "avg unique rates"
FROM royalties;
```

rates sum	sum unique rates	rates avg	avg unique rates
0.71	0.55	0.054615	0.061111

2.6. Referencias y material adicional

- Java Code Junkie (2020, September 27). *What is SQL?* [in 4 minutes for beginners]: https://www.youtube.com/watch?v=27axs9dO7AE
- Java Code Junkie (n.d). *MariaDB Tutorial for Beginners:* https://www.youtube.com/watch?v=-ARMty_N0RU
- N.D. *How to install MariaDB and HeidiSQL on Windows?* https://www.youtube.com/watch?v=YVDoAZBfzSg
- Baldwin, C. (2023). *How to write basic SQL statements (Select, From):* https://www.youtube.com/watch?v=YfTDBA45PHk
- Baldwin, C. (2023). *Where clause SQL:* https://www.youtube.com/watch?v=Cb0vCcUugYo
- Finer, A. (2021). *Learn basic SQL in 15 minutes (Part 2/3):* https://www.youtube.com/watch?v=gm6tNK_iOHs&t=0s
- Finer, A. (2022). *Learn basic SQL in 15 minutes (Part 3/3):* https://www.youtube.com/watch?v=w3ea4fKiS2g&t=0s
- Finer, A. (2021). *SQL Joins: Difference between Inner/Left/Right/Outer Joins:* https://www.youtube.com/watch?v=zGSv0VaOtR0
- Finer, A (2018*). What are pivot tables?* https://www.youtube.com/watch?v=wHmTDXrpsDc
- Blue, J. (2018). *SQL Joins tutorial for beginners: Inner Join, Left Join, Right Join, Full Outer Join*: https://www.youtube.com/watch?v=2HVMiPPuPIM
- Keith, M. (2021). SQL: *Is NULL, is NOT NULL:* https://www.youtube.com/watch?v=8cI-QcaaKis
- N.D. (2022). *SQL Query on multiple tables*: https://www.youtube.com/watch?v=KmuPGDZ_Pyo
- Glitz, C. (2023). *Join multiple tables in SQL:*

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

- Kanthety,S.S. (2023). *GROUP BY and HAVING clause in SELECT Command:* https://www.youtube.com/watch?v=wsbp1J-AFhY
- N.D. (2021). *Difference between WHERE and HAVING clause:* https://www.youtube.com/watch?v=w2TSylcxxig
- N.D. (2021). *Advanced SQL Tutorials/Subqueries*: https://www.youtube.com/watch?v=m1KcNV-Zhmc
- N.D. (2019). *How to use the BETWEEN condition in SQL*: https://www.youtube.com/watch?v=oiOvdZudb_I
- Baldwin, C. (2023). *Working with Dates (SQL):* https://www.youtube.com/watch?v=XyZ9HwXoR7o
- Baldwin, C. (2023). *Cleaning Strings (SQL) – LEFT, RIGHT, TRIM, UPPER, LOWER, CONCAT:* https://www.youtube.com/watch?v=1Kx_VjCtuFU
- Baldwin, C. (2023). SQL Query Optimization-Tips for more efficient Queries: https://www.youtube.com/watch?v=GA8SaXDLdsY

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132

Co-funded by the European Union

GRANT AGREEMENT 2021-1-ES01-KA220-VET-000033132