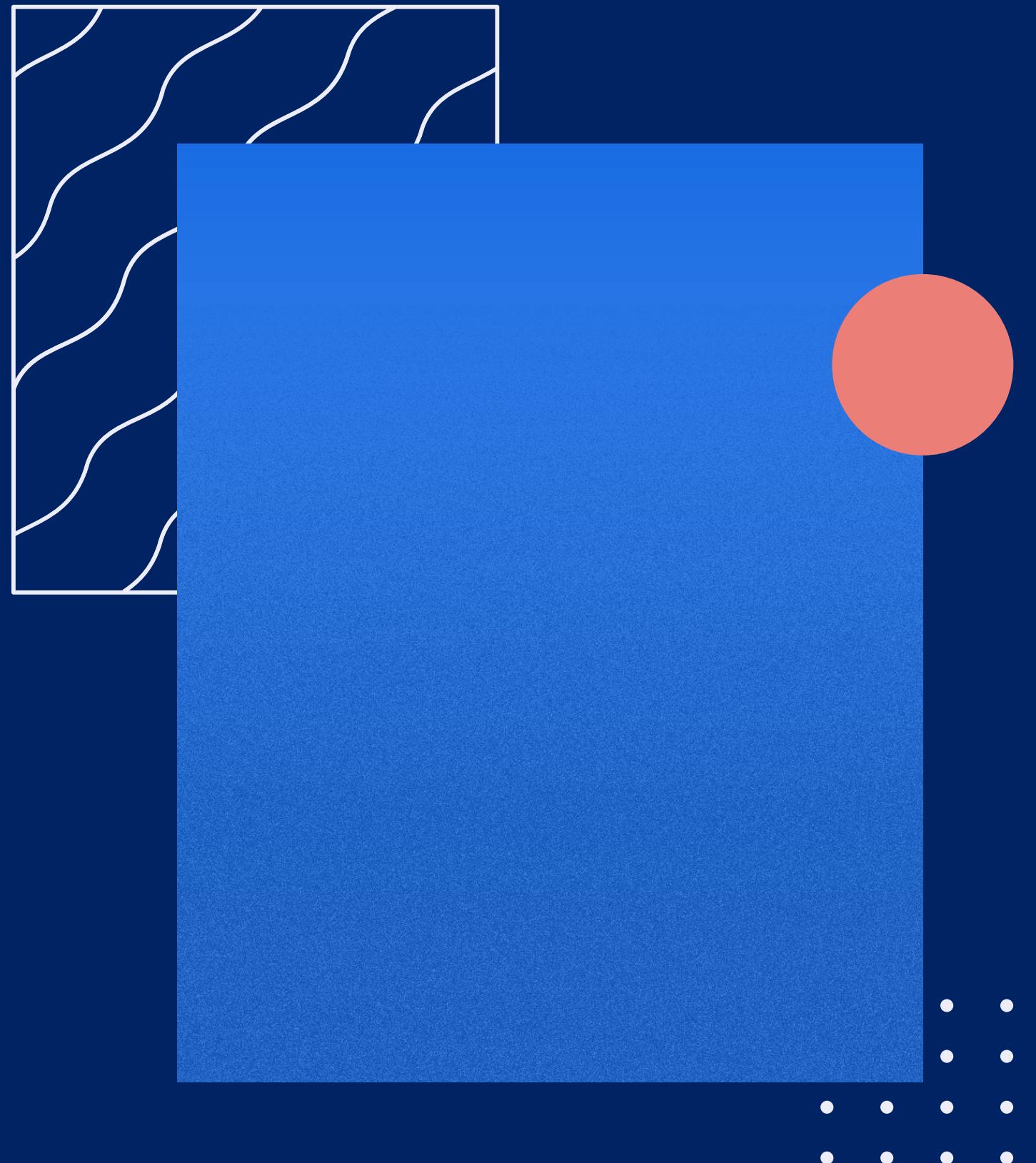


Introducción a Bases de Datos y SQL

MODULO 2 - OBJETIVOS

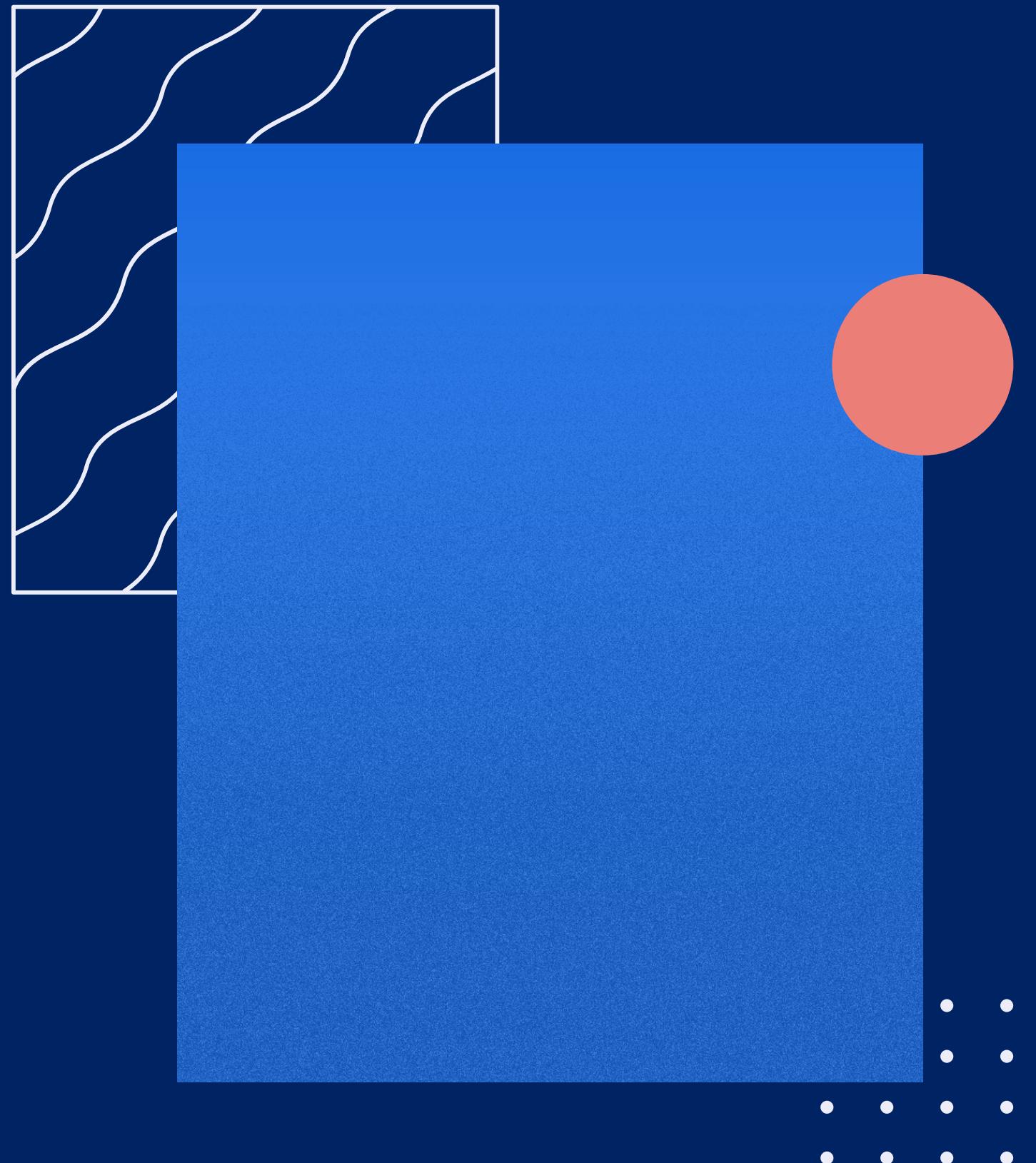


MODULO 2 - OBJETIVOS

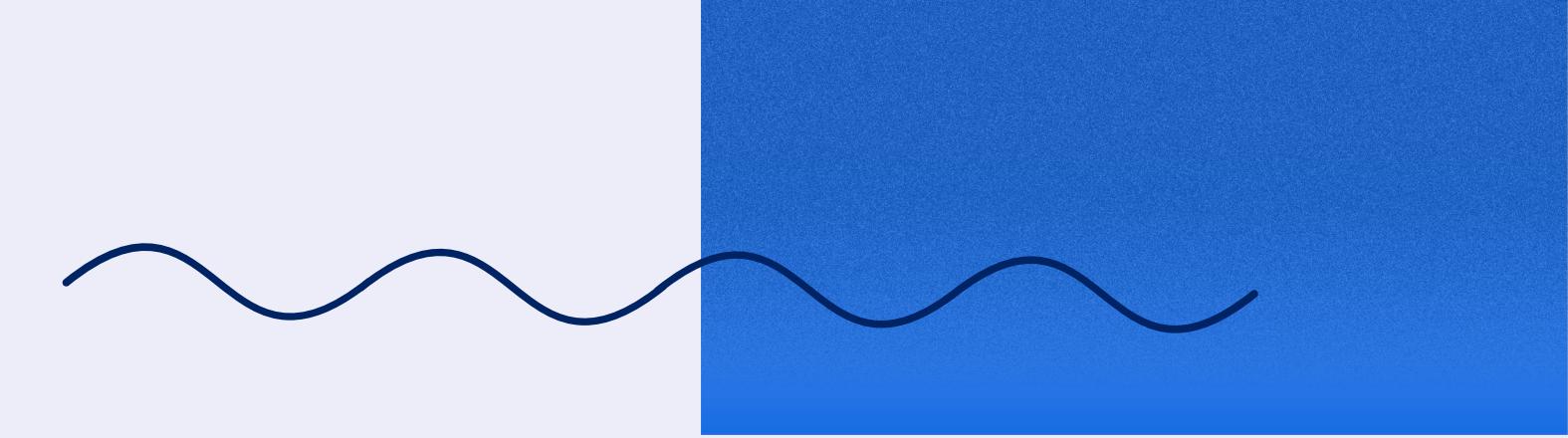
1. Importar tablas externas.
2. Estructura del lenguaje SQL.
3. Generar consultas utilizando lenguaje SQL.
4. Ordenamiento de registros.
5. Limitar la cantidad de registros resultantes en una consulta.
6. Utilizar predicados en consultas (cláusula WHERE).
7. Filtrar datos de una columna de acuerdo a condiciones múltiples en una consulta, utilizando operadores de comparación y operadores lógicos.

Introducción a Bases de Datos y SQL

MODULO 2



IMPORTACIÓN DE TABLAS EXTERNAS



IMPORTAR TABLAS DESDE ORÍGENES EXTERNOS

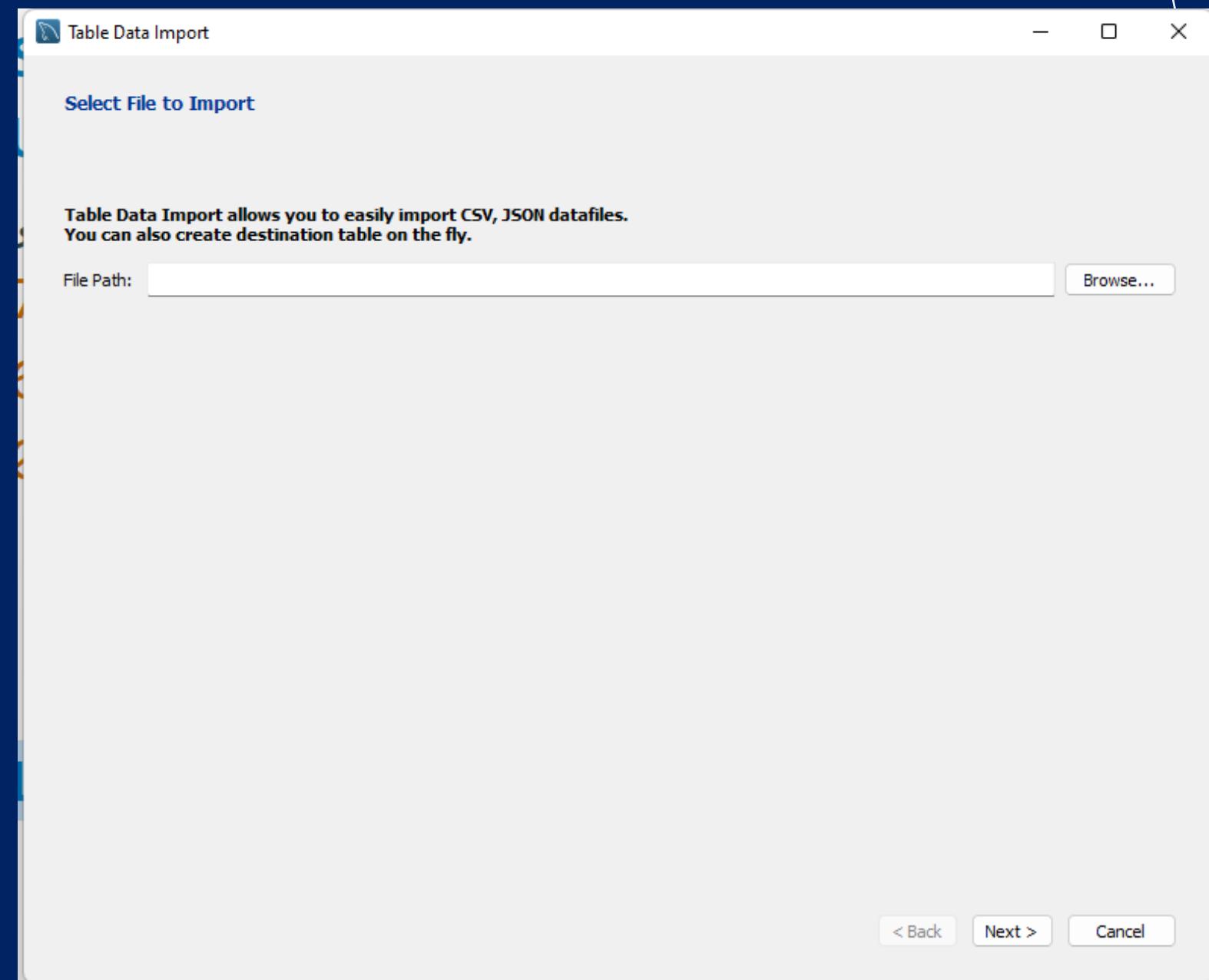
En muchas oportunidades se presenta el caso de que una empresa posee las tablas generadas en Microsoft Access o Microsoft Excel y debido a la cantidad de información que empiezan a contener esas tablas, se decide pasar las tablas a SQL para trabajarlas de una mejor manera. MySQL Workbench permite la importación de tablas externas. El único problema que se presenta es que este motor de bases de datos sólo admite la importación de tablas que se hallen en archivos con formato CSV o JSON.

Dado que esas tablas no fueron generadas dentro de un entorno SQL, los campos no tendrán definidos tipos de datos ni modificadores. Al momento de llevar a cabo la importación de las tablas, se podrá definir el tipo de dato para cada uno de los campos.

Mecanismo de importación

Para importar tablas desde archivos con formato CSV o JSON:

1. Pulsa el clic derecho del mouse sobre el nombre de la base de datos en uso.
2. Selecciona la opción Table Data Import Wizard. Al seleccionar esta opción se mostrará la siguiente pantalla

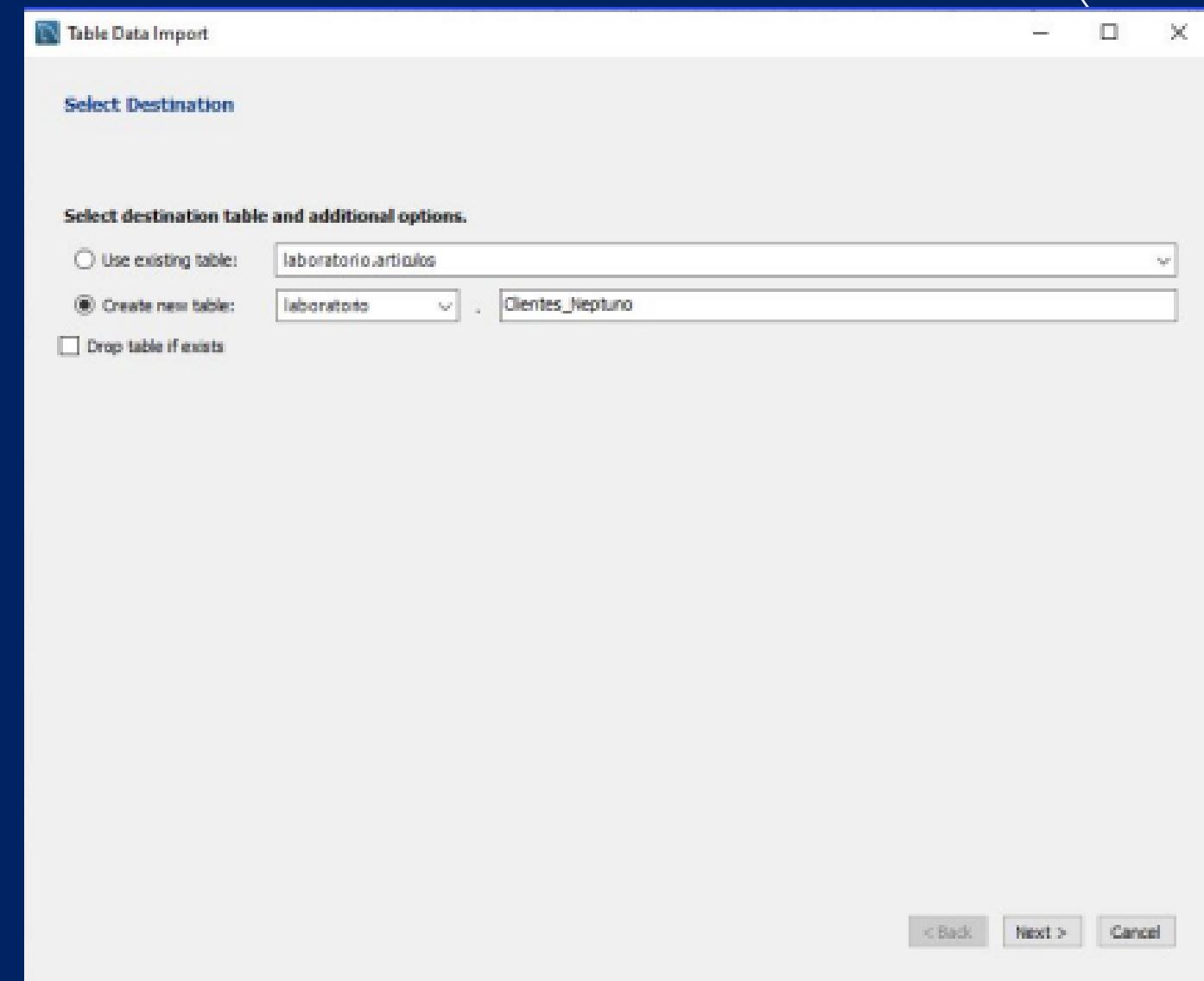


3. Pulsa el botón Browse.

4. Ubica y abre el archivo que contiene la tabla a importar.

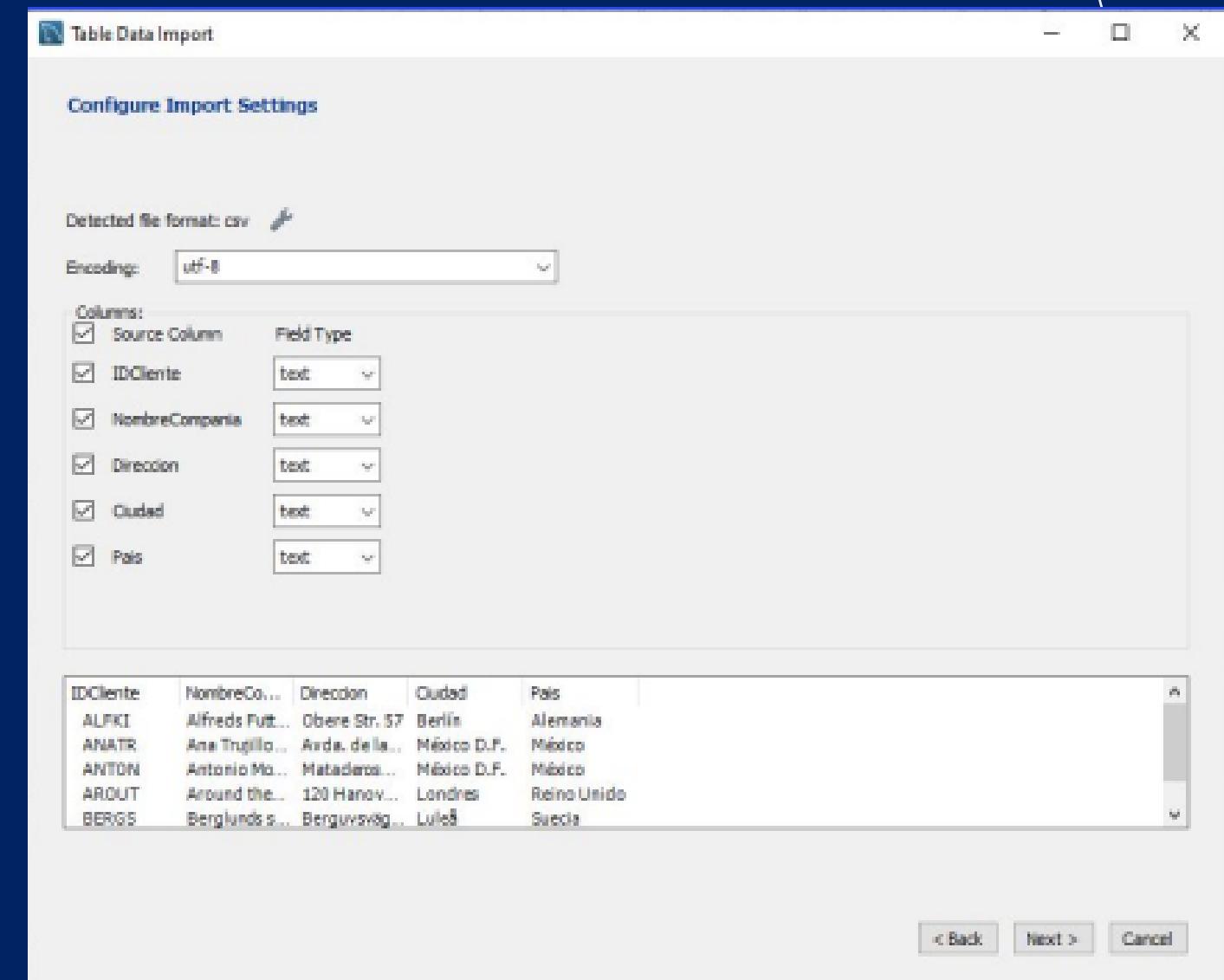
5. Pulsa el botón Next.

6. Especifica si quieres volcar los datos de la tabla importada en una tabla existente dentro de la base de datos, o selecciona la opción Create new table para crear una nueva tabla dentro de la base de datos. Puedes cambiar el nombre de la tabla, o mantener el nombre por defecto (el nombre del archivo). Pulsa Next

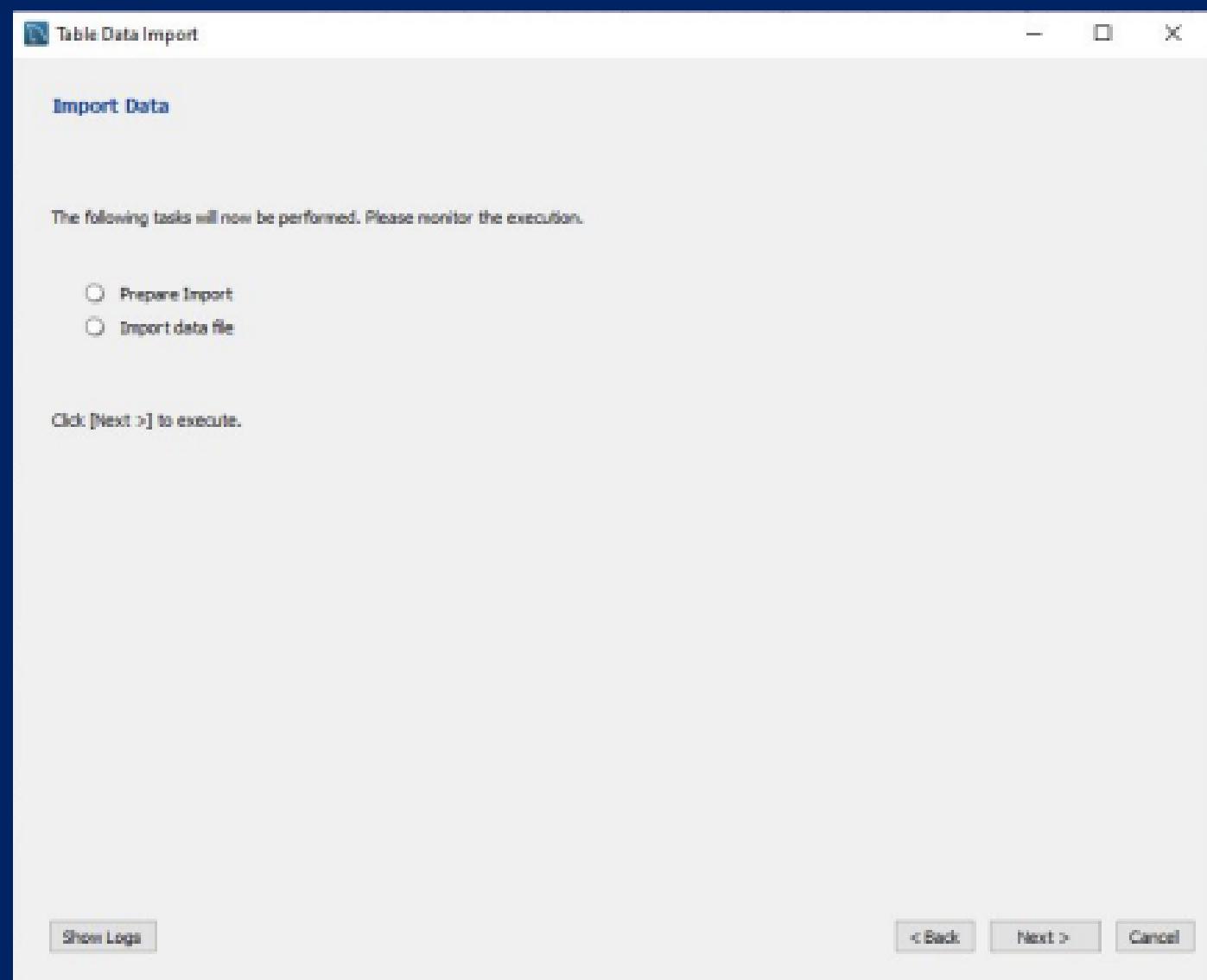


7. En este paso, podrás especificar los tipos de dato para cada campo. Por defecto, MySQL Workbench asignará a cada campo (columna) un tipo de dato según los valores que encuentre en ellos. Estos tipos de datos asignados se pueden cambiar, seleccionándolos de las listas desplegables.

8. Una vez definidos los tipos de datos para campo, pulsa Next.



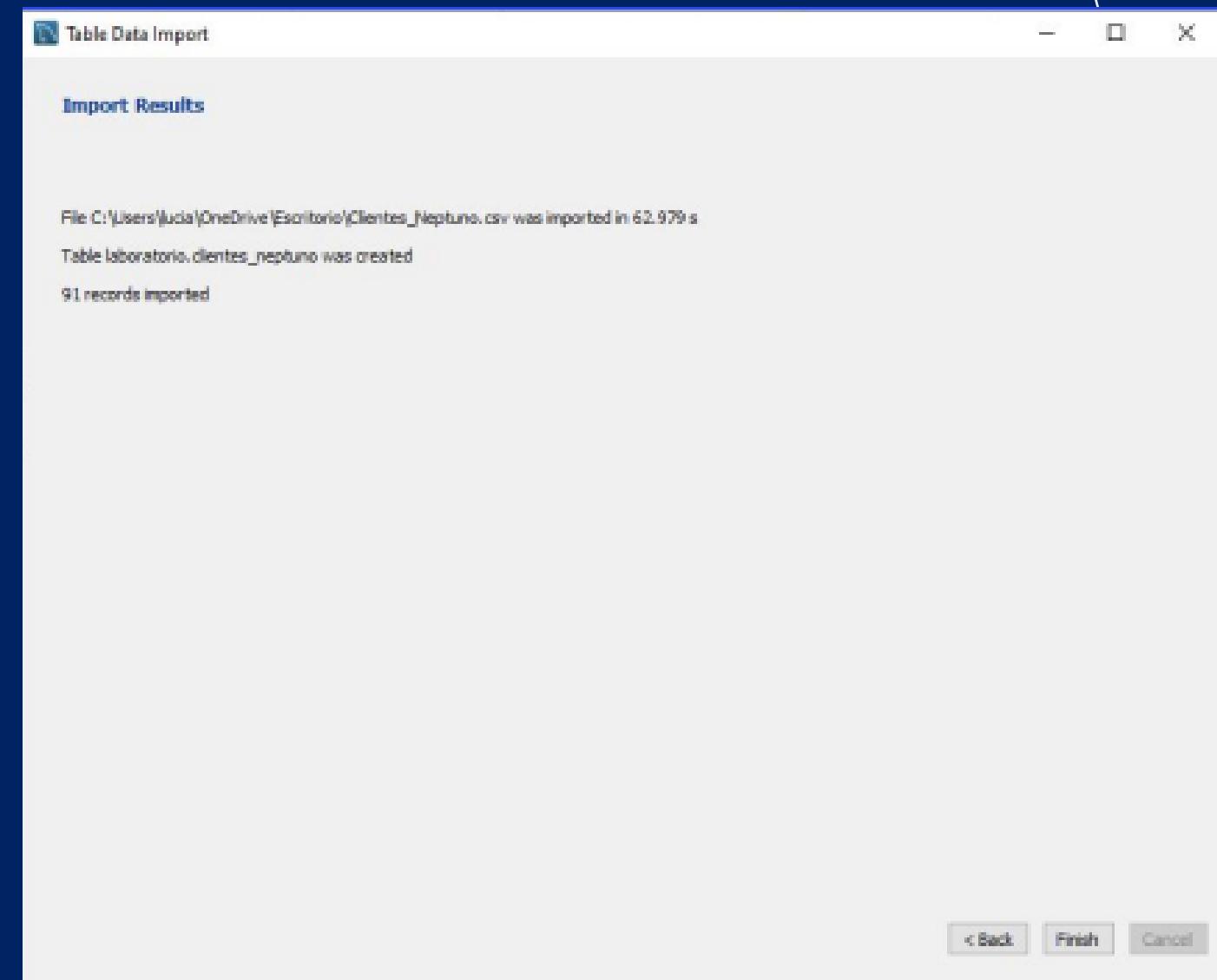
9. En este paso, sólo se especifica que se llevará a cabo la importación de los datos desde el archivo de datos externo. Sólo bastará con pulsar Next para comenzar la importación.



10. Finalizado el proceso de importación, pulsa el botón Next para poder observar la pantalla final. En esta vista encontrarás el tiempo que tardó la importación de los datos, la confirmación de que la tabla fue creada dentro de la base de datos y la cantidad de registros importados desde el origen externo.

11. Pulsa Finish para concluir el asistente.

12. No olvides actualizar los esquemas para observar la tabla dentro de la base.

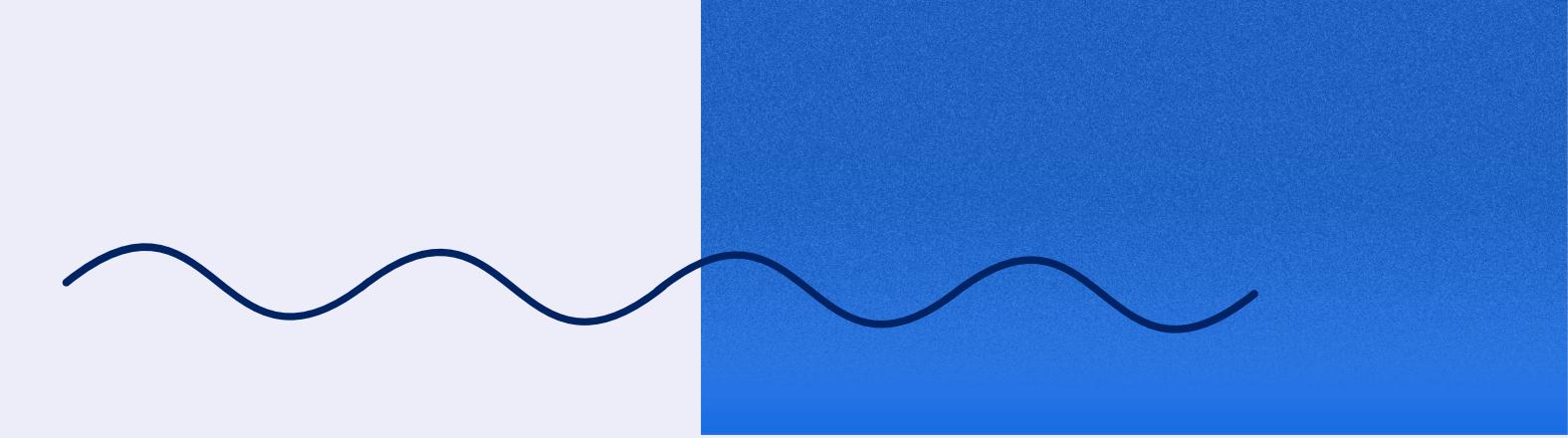


GENERAR TABLAS DESDE SCRIPTS

Generar tablas desde scripts En el caso de contar con un archivo SQL que contenga un script que genera una tabla, podrás abrirlo desde el motor de base de datos y ejecutarlo para crear la tabla dentro de la base de datos en uso.



GENERAR TABLAS DESDE SCRIPTS SQL

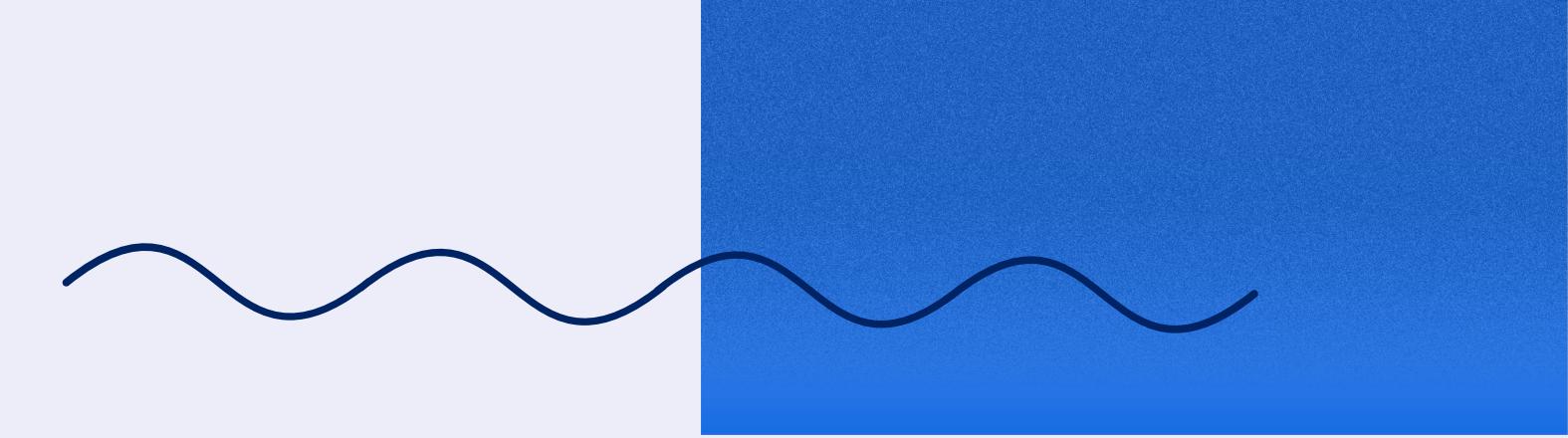
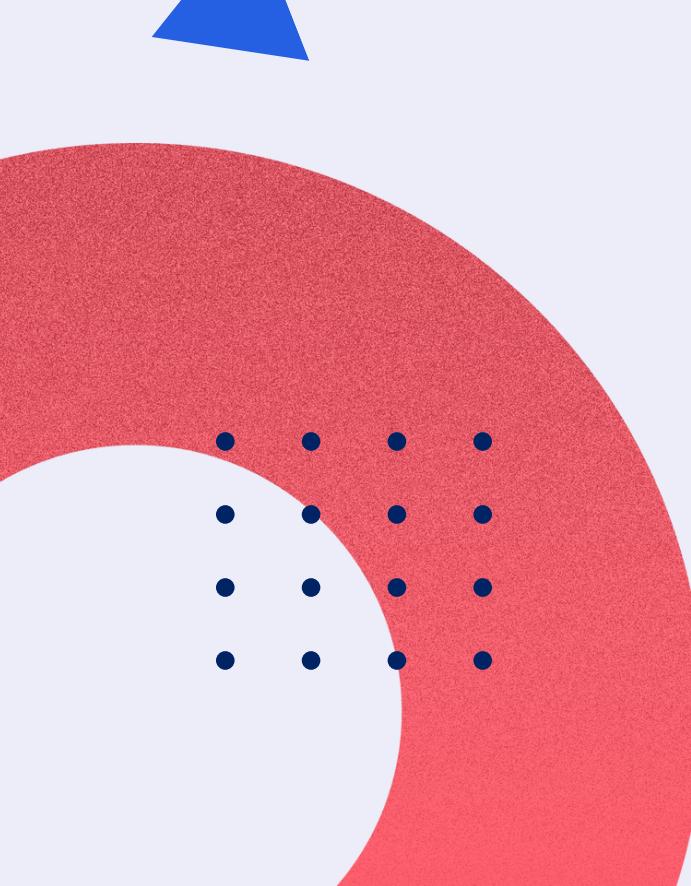


Los pasos para lograrlo son:

1. Abre MySQL Workbench.
2. Ejecuta el comando File Open SQL Script.
3. Ubica y abre el archivo SQL que contenga el script que genera la tabla.
4. Ejecuta el script.
5. Actualiza los esquemas.



CONSULTAS DE LENGUAJE SQL



SELECCIONAR REGISTROS DE UNA TABLA: CLÁUSULAS SELECT

La sentencia SELECT permite realizar operaciones de selección, ordenación, agrupación y filtrado de registros. Esta instrucción o sentencia utiliza diversas cláusulas:

FROM	WHERE	GROUP BY	HAVING	ORDER BY
Especifica la tabla de la que se quieren obtener los registros	Especifica los criterios o condiciones que deben cumplir los registros a buscar dentro de la tabla	Permite agrupar los registros seleccionados en función de uno o más campos	Especifica las condiciones que deben cumplir los grupos generados	Ordena los registros seleccionados en función de un campo

Mostrar todo el contenido de una tabla

En el ejemplo de la derecha, se *selecciona de la tabla Articulos todos los registros contenidos en la tabla y se muestran todas las columnas.*

El asterisco (*) ubicado a continuación de la sentencia SELECT especifica que, en el resultado de la consulta, se deben *mostrar todas las columnas (campos) contenidos en la tabla.*

SELECT * FROM Usuarios;

Mostrar algunos campos de una tabla

En el siguiente ejemplo, se selecciona de la tabla Articulos los valores de la columna Nombre, de todos los registros contenidos en la tabla:

```
SELECT Nombre FROM Usuarios;
```

Y en el siguiente ejemplo, se selecciona de la tabla Articulos *todos los valores de las columnas Nombre y Precio*, de todos los registros contenidos en la tabla:

```
SELECT Nombre, Precio FROM Usuarios;
```

Generar columnas en una consulta

En el siguiente ejemplo, se selecciona de la tabla Articulos los valores de todas las columnas y se agrega una nueva columna con el nombre *Precio con Aumento*, al *incrementar en un 25% el valor de la columna Precio*:

```
SELECT *, Precio * 1.25 as 'Precio con Aumento' FROM Articulos;
```



Y en el siguiente ejemplo, se selecciona de la tabla Articulos los valores de todas las columnas y agrega una nueva columna con el nombre *Origen*, con el valor constante *China*:

```
SELECT *, 'China' as Origen FROM Articulos;
```

Ordenamiento de datos: ORDER BY

La cláusula *ORDER BY* tiene como finalidad ordenar los resultados de las consultas por otras columnas. Cuando se genera un *SELECT* en base a una tabla, el resultado siempre se muestra ordenado en base al campo índice (*PRIMARY KEY*) por defecto. Tenga en cuenta que:

- El orden puede ser ascendente (por defecto) o descendente.
- Se puede ordenar por una columna o un conjunto de ellas.



En el siguiente ejemplo, se ordena el resultado de la consulta por el campo apellido (por defecto, en orden ascendente):

```
SELECT nombre, apellido FROM clientes ORDER BY apellido; -- ASC
```

Y en el siguiente ejemplo, se ordena el resultado de la consulta por el campo apellido (por defecto, en orden ascendente) y por el campo nombre en orden descendente:

```
SELECT nombre, apellido FROM clientes ORDER BY apellido, nombre DESC;
```

Limitar la cantidad de registros

LIMIT

Para poder *limitar el número de filas* (registros/resultados) devueltas en las consulta generada a través de la sentencia *SELECT*, se debe utilizar la cláusula *LIMIT*.

A su vez, esta cláusula permite establecer el número máximo de registros a eliminar en el caso de utilizar la sentencia *DELETE*.

En el ejemplo siguiente, se limita a 2 (dos) el número total de registros a mostrar en el resultado de la consulta:

```
SELECT nombre, apellido  
FROM clientes  
LIMIT 2;
```

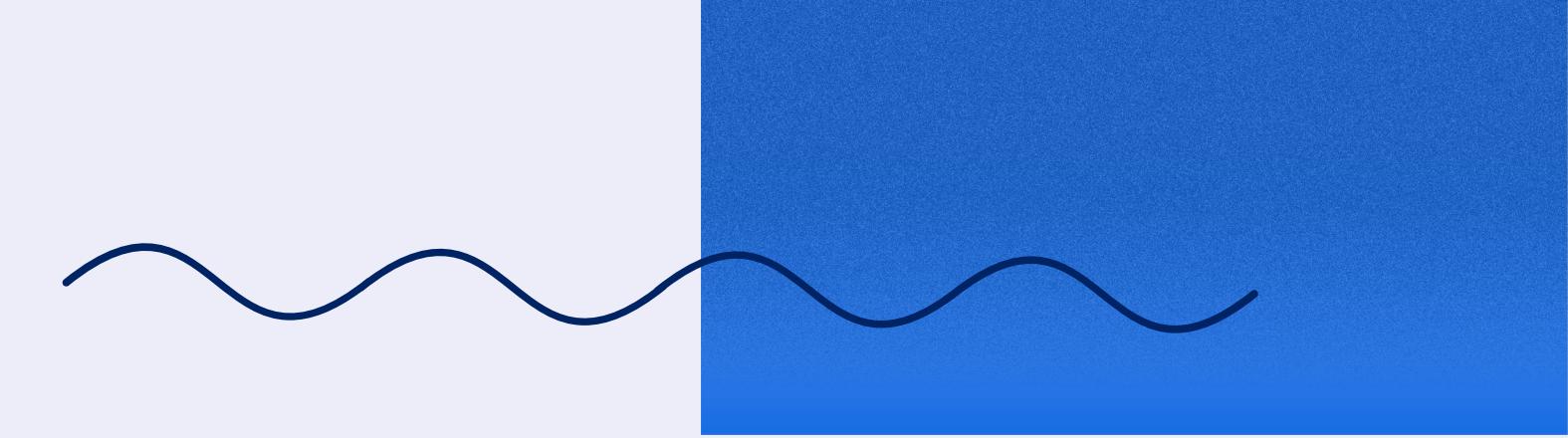
OFFSET

*Opera en combinación con la cláusula **LIMIT** y permite posicionarse en el registro indicado; **LIMIT** seleccionará la cantidad de registros establecidos en la sentencia. En el ejemplo de la derecha, se limita a 2 (dos) el número total de registros a mostrar en el resultado de la consulta, a partir del 5º (quinto) registro (es decir, saltando los 4 primeros):*

**SELECT nombre, apellido
FROM clientes
LIMIT 2 OFFSET 4;**

En el ejemplo siguiente, se limita a 2 (dos) el número total de registros a mostrar en el resultado de la consulta:

PREDICADO DE CONSULTA SQL



CLÁUSULA WHERE

La cláusula *WHERE* permite especificar las *condiciones o criterios que deben cumplir los registros* a buscar dentro de una tabla. Recuerda siempre el orden en que se deben especificar las cláusulas:

FROM	WHERE	GROUP BY	HAVING	ORDER BY

Al especificar *condiciones* de búsqueda dentro de una tabla, se utilizará la cláusula WHERE, la cual se debe colocar obligatoriamente después de la cláusula FROM.

OPERADORES DE COMPARACIÓN

Operador	Descripción
=	Igual a
<	Menor que
>	Mayor que
>=	Mayor o igual que
<=	Menor o igual que
<>	No es igual a



En el siguiente ejemplo, se selecciona de la tabla Articulos la columna *Nombre* y muestra todos aquellos registros cuyo valor en la columna *codigo* sea igual a 1:

```
SELECT Nombre FROM Articulos WHERE codigo = 1;
```

Y en el siguiente ejemplo, se selecciona de la tabla Articulos las columnas *Nombre* y *Precio* y muestra todos aquellos registros cuyo precio sea superior a 150:

```
SELECT Nombre, Precio FROM Articulos WHERE Precio > 150;
```

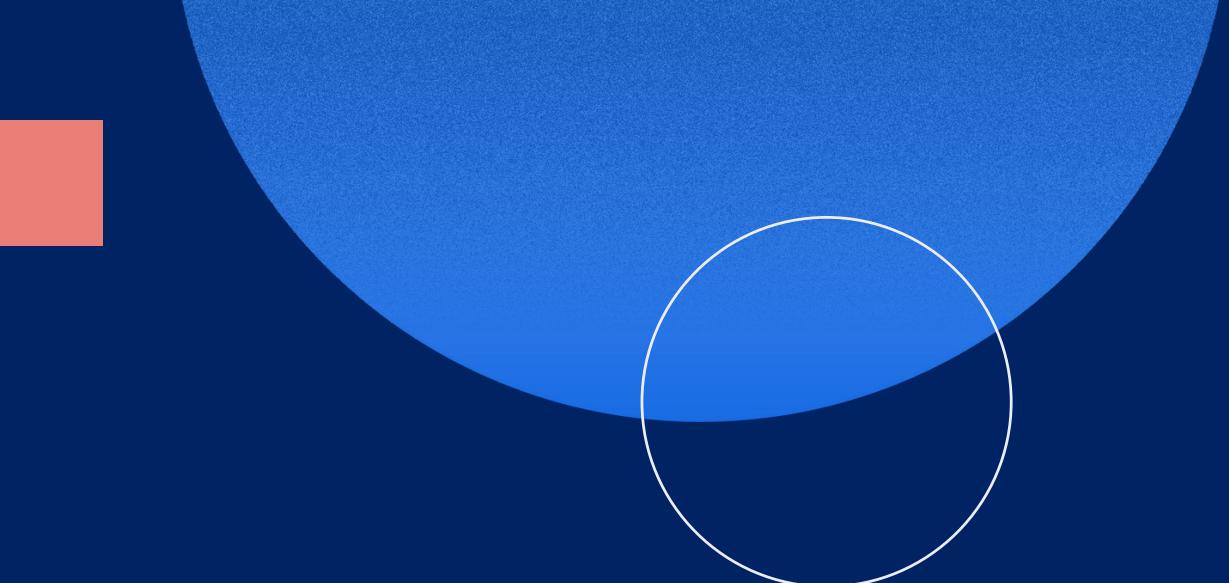
Operadores lógicos

Para crear expresiones lógicas disponemos de varios operadores de comparación.

Estos operadores se aplican a cualquier tipo de columna (fechas, cadenas, números, etc.). Y devuelven valores lógicos, que son: verdadero o falso (1 ó 0).

Operador	Descripción
AND	Se deben cumplir todas las condiciones especificadas
OR	Se debe cumplir al menos una de las condiciones especificadas
NOT	No debe cumplir las condiciones especificadas

- Si uno o los dos valores a comparar son NULL, el resultado es NULL. (Excepto con el operador $<=>$ que es usado para una comparación con NULL segura).
- El operador $<=>$ funciona igual que el operador $=$. Salvo que, si en la comparación una o ambas de las expresiones es nula, el resultado no es NULL. Si se comparan dos expresiones nulas, el resultado es verdadero.



En el siguiente ejemplo, se selecciona de la tabla Articulos todos aquellos registros cuyo precio tenga un valor mayor o igual a 500, o su stock sea mayor o igual a 100:

```
SELECT * FROM Articulos WHERE precio >= 500 OR stock >= 100;
```

Y en el siguiente ejemplo, se selecciona de la tabla Articulos *todos los valores de las columnas Nombre y Precio*, de todos los registros contenidos en la tabla:

```
SELECT * FROM Articulos WHERE Precio < 20 AND stock >= 100;
```

Operadores BETWEEN / NOT BETWEEN

Entre los operadores de MySQL, existe uno denominado BETWEEN (entre), el cual se utiliza para comprobar si una expresión está comprendida en un determinado rango de valores. La sintaxis es:

- BETWEEN mínimo AND máximo

```
SELECT * FROM Articulos WHERE precio BETWEEN 100 AND 200;
```

- NOT BETWEEN mínimo AND máximo

```
SELECT * FROM Articulos WHERE precio NOT BETWEEN 100 AND 200;
```

Operadores IN / NOT IN

Los operadores IN y NOT IN sirven para averiguar si el valor de una expresión determinada se encuentra dentro de un conjunto indicado. Su sintaxis es:

Su sintaxis es:

- IN (<expr1>, <expr2>, <expr3>,...)
- NOT IN (<expr1>, <expr2>, <expr3>,...)

- El operador IN devuelve un valor verdadero si el valor de la expresión es igual a alguno de los valores especificados en la lista.
- El operador NOT IN devuelve un valor falso en el caso contrario.



Ejemplo 1

```
SELECT * FROM Articulos WHERE codigo IN (1,2,3);
```

Ejemplo 2

```
SELECT * FROM Articulos WHERE codigo IN (1,2,3);
```

Ejemplo 3

```
SELECT * FROM Articulos WHERE nombre NOT IN ('Pala', 'Maza');
```

Operadores *LIKE / NOT LIKE*

El operador **LIKE** se usa para hacer *comparaciones entre cadenas y patrones*.

- El resultado es verdadero (1) si la cadena se ajusta al patrón y falso (0) en caso contrario.
- Tanto si la cadena como el patrón son **NULL**, el resultado es **NULL**.

Para definir estos patrones, se hace uso de comodines, como vemos en el cuadro de la derecha.

Y en el ejemplo, se muestra de la tabla Articulos todos aquellos registros que en el campo nombre, figure la palabra Pala:

- El operador **IN** devuelve un valor verdadero si el valor de la expresión es igual a alguno de los valores especificados en la lista.
- El operador **NOT IN** devuelve un valor falso en el caso contrario.

```
SELECT * FROM Articulos WHERE nombre  
LIKE '%Pala%';
```



Como siempre que se usan caracteres concretos para crear patrones, se presenta la dificultad de hacer comparaciones cuando se deben buscar precisamente esos caracteres concretos.

La comparación es independiente del tipo de los caracteres. Es decir, LIKE no distingue mayúsculas de minúsculas, salvo que se indique lo contrario.

Secuencias de escape: '\'

La dificultad mencionada, se suele superar mediante secuencias de escape. Si no se especifica nada en contra, el carácter que se usa para escapar es '\'. De este modo, si queremos que nuestro patrón contenga los caracteres '%' o '_', los escaparemos de este modo: '\%' y '_'. Ejemplo:

```
SELECT * FROM clientes WHERE mail LIKE '%\_%';
```

Operadores IS NULL / IS NOT NULL

Los operadores IS NULL e IS NOT NULL sirven para verificar si una expresión determinada es o no nula.

En el siguiente ejemplo, se mostrará todos aquellos registros de la tabla clientes que no tengan cargado ningún valor en el campo comentarios:

```
SELECT * FROM clientes WHERE comentarios IS NULL;
```

PREGUNTAS???

