



# BASES DE DATOS RELACIONALES





## ¿QUÉ VAMOS A VER?

- Creando una base de datos relacional.
- Crear nuevo usuario.
- Consultando datos: SELECT.
- SELECT: Operaciones de filtrado.
- Establecer un Alias.
- Agrupaciones de datos.



# ¡VAMOS A COMENZAR!





# CREANDO UNA BASE DE DATOS RELACIONAL



# CONEXIÓN A LA BASE DE DATOS

- Por defecto, MySQL Workbench, nos genera una instancia local de MySQL80.
- Esta conexión la podemos ocupar tranquilamente para manejar nuestros bases de datos local, ya que se accede a ella a través del usuario root, es decir, tenemos absoluto control sobre sus accesos y funcionalidades.
- Es importante tener en cuenta que, para poder acceder a la conexión por defecto, o generar una nueva propia, debemos tener a mano si o si la clave del usuario root que ingresamos al momento de instalar MySQL.

The screenshot shows the MySQL Workbench interface. At the top, there's a list titled 'MySQL Connections' with one entry: 'Local instance MySQL80'. Below it, a detailed view shows the connection parameters: 'root' user, 'localhost:3306' host, and a profile icon. Below this, a larger window titled 'Setup New Connection' is open. It has tabs for 'Parameters', 'SSL', and 'Advanced'. Under 'Parameters', the 'Hostname' is set to '127.0.0.1', 'Port' to '3306', 'Username' to 'root', and 'Default Schema' is empty. There are also sections for 'SSL' and 'Advanced' settings. At the bottom of the dialog are buttons for 'Configure Server Management...', 'Test Connection', 'Cancel', and 'OK'.



# CONVENCIÓN DE NOMBRES EN BBDD

- La recomendación inicial es que es ideal que todo nombre sea utilizando **inglés**.
- Los nombres de las tablas deben ir siempre en **mayúsculas** y en **plural**, por ejemplo. *USUARIOS* o *PERSONAS*. (Existen ocasiones que el RDBMS cambia las letras a minúsculas, en ese caso se respeta eso).
- Los nombres de los campos deben ir siempre en minúsculas y en singular. En caso de ser palabras compuestas, se deben separar con un guion bajo, por ejemplo. *tarjeta\_credito* o *nombre\_completo*.





# CREACIÓN DE LA BASE DE DATOS

- Los gestores de bases de datos, siempre nos van a dar la opción de poder crear un esquema a través de su interfaz gráfica, y de esa forma facilitarnos la vida.
- **¿Es lo mejor cuando estoy comenzando?**  
No realmente, ya que lo ideal es aprender a utilizar las consultas SQL y el mismo lenguaje en sí.
- CREATE DATABASE nombre\_bbdd;





# CREACIÓN DE CAMPOS

- Necesitamos tener un campo de la tabla que sea el identificador de la misma, es decir, la **primary key**.
- Cada uno de los campos que vamos a crear necesita tener un tipo de dato específico.
  - **INT**: números enteros.
  - **DOUBLE**: números decimales.
  - **VARCHAR()**: cadenas de texto. Dentro de los () se le pasa la cantidad máxima de caracteres que soporta.
  - **DATE**: formato de fecha.
- Nos vamos a encontrar con opciones importantes para cada campo, como **NN** (Not Null) y **AI** (Auto Incremental).

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id_usuario	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>						

Column Name: id\_usuario  
Charset/Collation: Default Charset  
Comments:

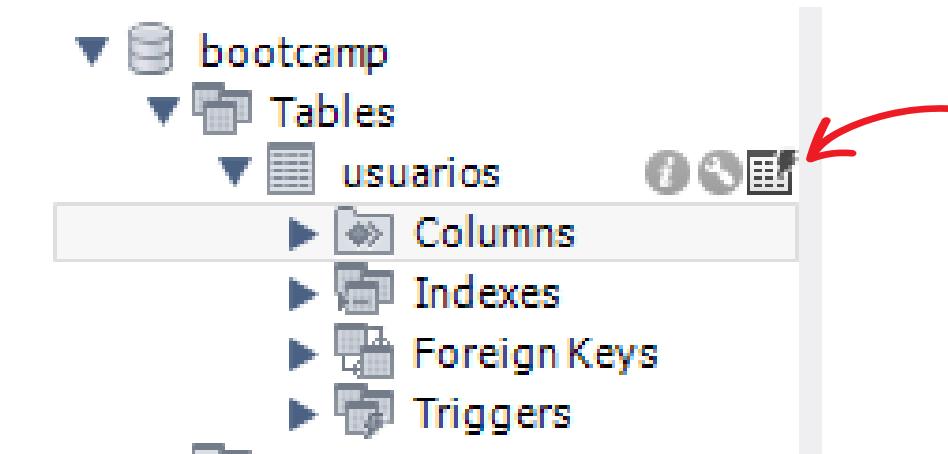
Datatype dropdown options (visible):

- INT
- INT
- VARCHAR()
- DECIMAL()
- DATETIME
- BLOB
- 
- BINARY()
- BLOB()
- LONGBLOB
- MEDIUMBLOB
- TINYBLOB
- VARBINARY()
- 
- DATE
- DATETIME()
- TIME()
- TIMESTAMP()
- YEAR()
- 
- GEOMETRY
- GEOMETRYCOLLECTION
- LINESTRING
- MULTILINESTRING
- MULTIPOINT
- MULTIPOLYGON
- POINT
- POLYGON
- 
- BIGINT()
- DECIMAL
- DOUBLE
- FLOAT
- INT()
- MEDIUMINT()
- REAL



# INGRESAR DATOS POR INTERFAZ

- Una vez tengamos creada la tabla, mediante el tercer ícono al lado de su nombre podremos acceder al menú gráfico que nos permite agregar nuevas filas con sus datos correspondientes.
- En este ejemplo, podemos ver en primera instancia de que existen campos nulos, y eso ocurre porque el modelo que utilizamos para crear la tabla.
- Manos al código, vamos a ver de qué manera el *id\_usuario* se vuelve auto incremental con la opción que le validamos.



	id_usuario	nombre	apellido	edad	fecha_inicial	email
▶	1	Carlos	Tapia	29	2018-10-05	ctapiadev@gmail.com
	2	Franco	Escamilla	43	1990-11-01	franco@gmail.com
	3	Maria	Gonzalez	NULL	NULL	maria@gmail.com
	4	Jonathan	Gutierrez	20	NULL	jona@gmail.com
	5	Francisca	Maira	23	NULL	fran@gmail.com
	6	Rafael	Sanchez	33	NULL	NULL
●	NULL	NULL	NULL	NULL	NULL	NULL



# CREAR NUEVO USUARIO



# CREAR NUEVO USUARIO

- MySQL Workbench nos da la opción de crear nuevos usuarios y otorgar privilegios por medio de su interfaz gráfica.
- Nos dirigimos a **Server -> Users and Privileges**
- Una vez dentro tenemos la opción de modificar los ya existentes o presionar en **Add Account**
- En la pestaña de **Administrative Roles** podemos otorgar diferentes permisos al usuario.
- Por ejemplo, si necesitamos que un usuario solo tenga acceso a la lectura de datos, debemos de seleccionar solo la casilla de **SELECT**

**Details for account newuser@%**

Login		Account Limits	Administrative Roles	Schema Privileges
Login Name:	newuser	You may create multiple accounts with the same name to connect from different hosts.		
Authentication Type:	Standard	For the standard password and/or host based authentication, select 'Standard'.		
Limit to Hosts Matching:	%	% and _ wildcards may be used		
Password:		Type a password to reset it.		
Confirm Password:		Consider using a password with 8 or more characters with mixed case letters, numbers and punctuation marks.		
		Enter password again to confirm.		
		Expire Password		



# CONSULTANDO DATOS: SELECT



## CONSULTA DE DATOS > SELECT

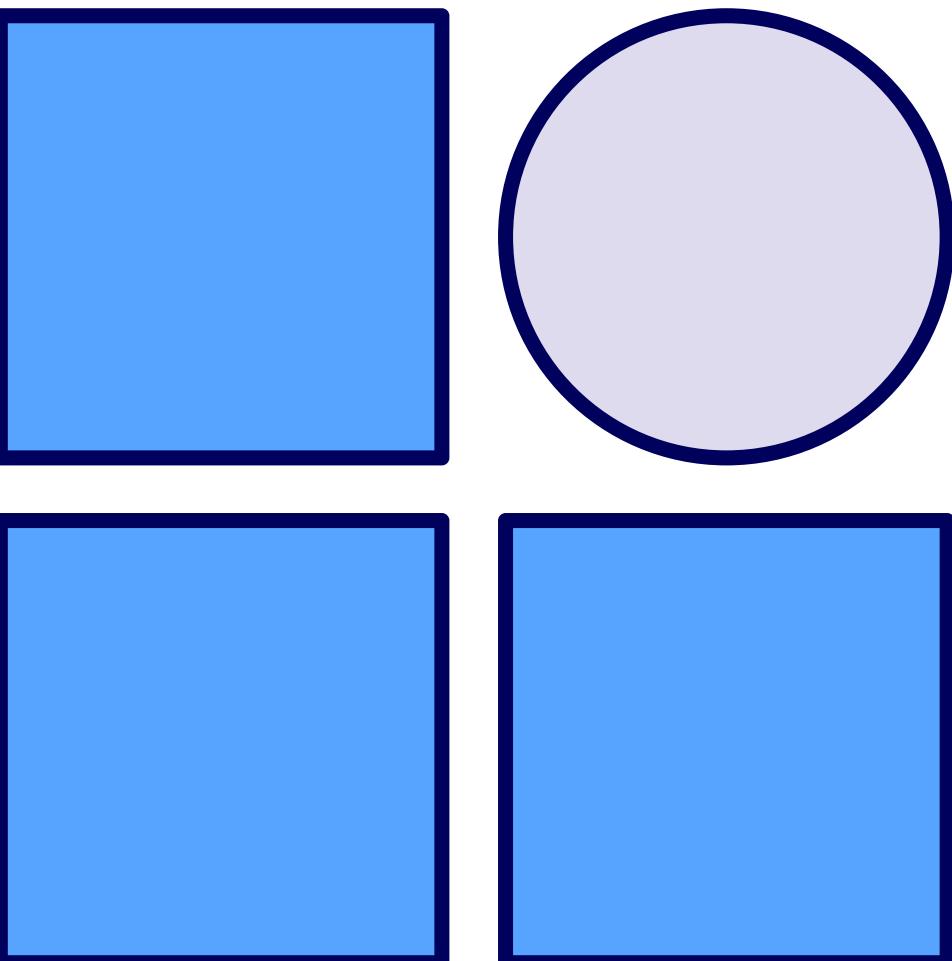
- La palabra reservada SELECT es la que vamos a utilizar constantemente para consultar los datos que están almacenados en una tabla.
- Además, debemos utilizar también la palabra reservada FROM para especificar el nombre de la tabla de la cual queremos obtener esos datos.
- **SELECT nombre\_columna FROM nombre\_tabla;**
- **¿Se puede cambiar el nombre de la columna por \*?** Sin ningún problema, la diferencia es que el símbolo representa a la totalidad de las columnas.





# CONSULTANDO DATOS DISTINTOS

- Dentro de la consulta de datos, también podemos decidir que solamente queremos obtener todos aquellos datos que son “distintos”, es decir, no necesito que mi consulta me retorne datos repetidos.
- Para eso, podemos utilizar la palabra reservada **DISTINCT** junto a un SELECT.
- `SELECT DISTINCT nombre_columna FROM nombre_tabla;`





# ¿Y SI BUSCAMOS EN BASE A UN CRITERIO?

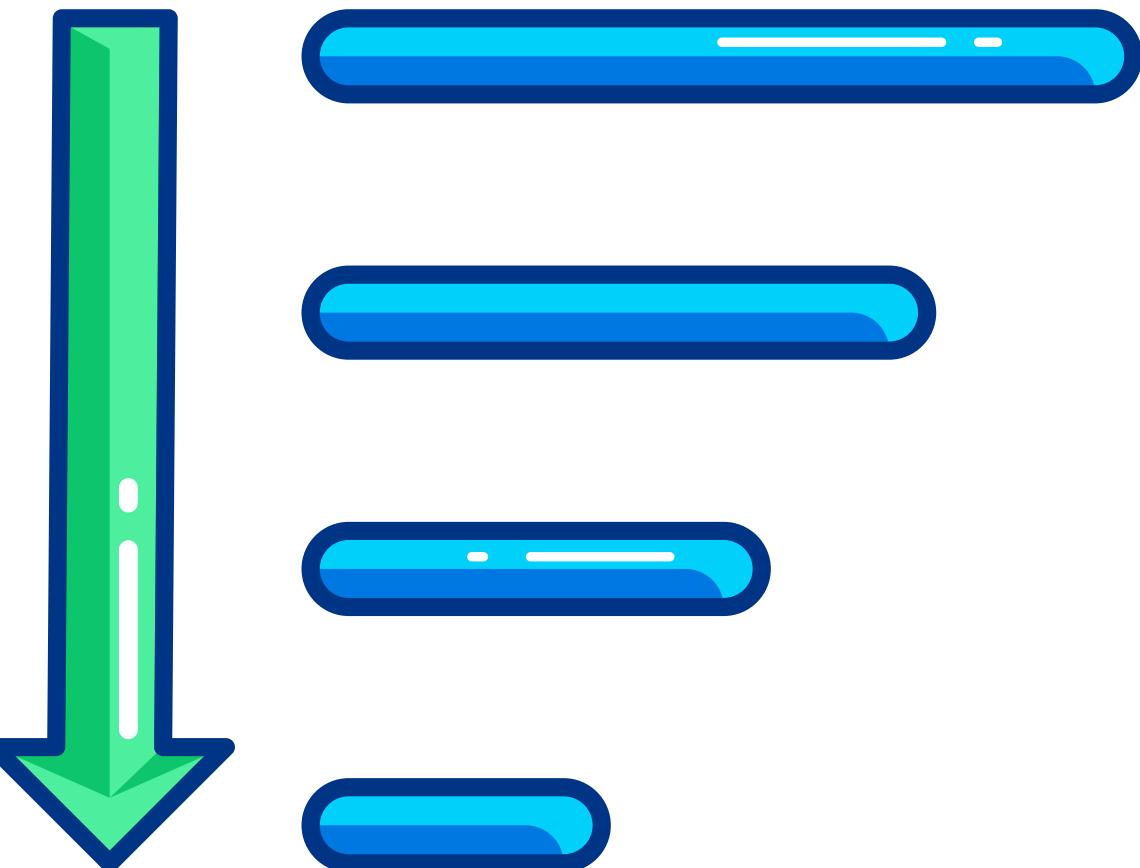
- Es en este punto de una consulta SELECT en dónde podemos aplicar algo más parecido a la lógica que aprendimos en los fundamentos de programación.
- Vamos a utilizar la palabra reservada **WHERE** para obtener una cierta cantidad de datos, basados en que se cumpla el criterio que le especifiquemos.
- *SELECT nombre\_columna FROM nombre\_tabla  
WHERE criterio;*





# ORDENAR DATOS CONSULTADOS

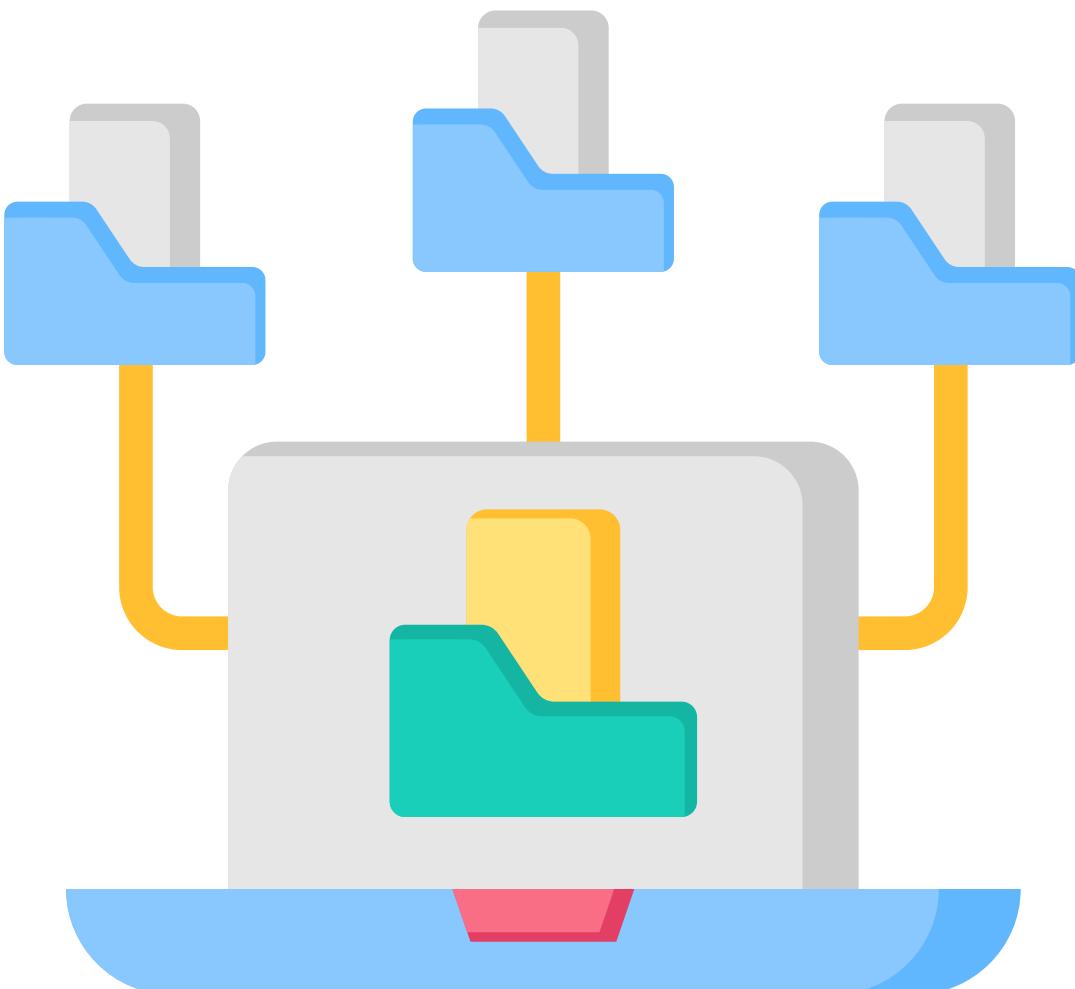
- Ya sabemos la forma correcta de obtener los datos que deseamos, pero ¿Cómo lo hacemos si queremos ordenarlos?
- Lo podemos hacer a través de la palabra reservada **ORDER BY** y sus sentencias que le acompañan **DESC** y **ASC** para especificar si quieres que nuestros datos se ordenen de manera ascendente o descendente.
- Por defecto los datos se ordenan de manera ascendente, en caso de que no especifiquemos la dirección.
- `SELECT nombre_columna FROM nombre_tabla  
ORDER BY campo DESC;`





# CONSULTANDO UN DATO CON UN PATRÓN

- No siempre vamos a estar seguros de lo que estamos buscando o requerimos una gama más amplia de datos que tienen un patrón en común.
- La palabra reservada LIKE a pesar de que ocupa a través de WHERE, la vamos a utilizar encontrar todos aquellos datos que cumplan con el patrón que le especifiquemos.
- `SELECT nombre_columna FROM nombre_tabla WHERE condición LIKE patrón`
- Para completar el dato que requiere nuestro patrón, vamos a utilizar el símbolo reservado %.





# CONSULTAS CONDICIONALES > AND, OR, NOT

- La cantidad de consultas que podemos aplicar sobre una tabla realmente no es una amplia gama de oportunidades para conseguir un dato.
- Esto se vuelve aún más dinámica cuando nos damos cuenta de que podemos incluir condiciones muy parecidas a las de la programación, para obtener un dato u otro.
- `SELECT nombre_columna FROM nombre_tabla WHERE condición AND/OR condición;`
- La palabra reservada NOT podemos agregarla luego de algún WHERE o AND/OR dependiendo del enfoque de nuestra consulta.





# CANTIDAD DE DATOS LIMITADOS

- En ocasiones, vamos a necesitar recibir una cantidad de datos limitados a partir de nuestra consulta.
- Para conseguirlo, debemos utilizar la palabra reservada **LIMIT** acompañada de un número entero que va a corresponder a la cantidad de datos que vamos a utilizar.
- `SELECT nombre_columna FROM nombre_tabla  
LIMIT 10;`





# SELECT: OPERACIONES DE FILTRADO



# FILTRAR DATOS EN CÁLCULOS DIRECTOS

- Los lenguajes de programación nos entregan una amplia gama de estructuras para nosotros realizar diferentes tipos de cálculos sobre datos concretos, pero... ¿Qué pasaría si las consultas SQL lo pueden hacer por nosotros?
- **MAX(column)** = Nos retorna el número máximo que encontramos dentro de una columna.
- **MIN(column)** = Nos retorna el número mínimo que encontramos dentro de una columna.
- **COUNT(column)** = Cuenta la cantidad de datos almacenados en una columna.
- **SUM(column)** = retorna la suma total de todos los datos de una columna.
- **AVG(column)** = retorna el promedio de todos los datos de una columna.



# FILTRAR UN DATO DIRECTO

- Ya tenemos las herramientas para buscar un dato que queremos en específico, pero que pasa si es más de un dato, o simplemente nuestra condicional no está muy clara respecto a la sensibilidad del tipo.
- Podemos utilizar la palabra reservada **IN()** para encontrar datos dentro de una tabla, siendo uno o más que vamos a separar por comas.
- `SELECT nombre_columna FROM nombre_tabla WHERE nombre_columna IN('dato1', 'dato2');`





# FILTRAR DATOS ENTRE UN VALOR Y OTRO

- A pesar de que esto lo podemos conseguir con condicionales sin ningún problema, tenemos a la mano un método de filtrado mucho más eficiente.
- La palabra reservada **BETWEEN** la vamos a utilizar para crear un rango entre dos datos números, y así recibir todos los datos que se encuentren en medio.
- *SELECT nombre\_columna FROM nombre\_tabla WHERE nombre\_columna **BETWEEN** num\_min **AND** num\_max;*





# ESTABLECER UN ALIAS



# ESTABLECER UN ALIAS A LAS COLUMNAS

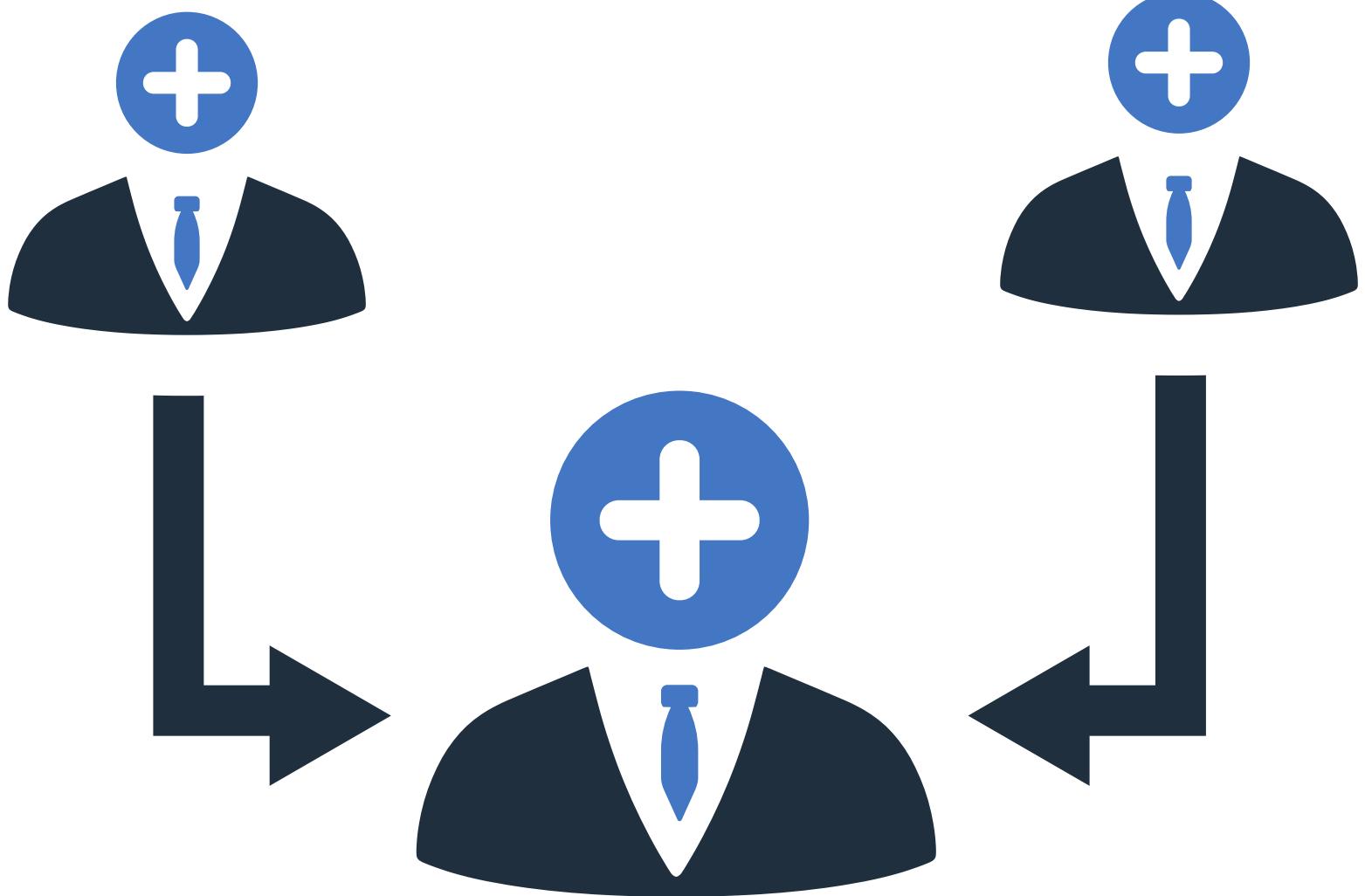
- ¿Es realmente necesario manejar un alias?
- Depende es la respuesta más acertada, porque todo va a depender de si estamos o no conformes con el nombre que representa nuestra columna, quizás no es lo óptimo para la manera en que queremos mostrar los datos, entonces en ese caso sí sería necesario hacer un pequeño cambio.
- Lo vamos a hacer por medio de la palabra reservada AS luego del nombre de la columna.
- `SELECT nombre_columna AS 'nuevo_nombre' FROM nombre_tabla;`
- Es importante tener en cuenta que es un cambio visual, no afecta al nombre de la columna que ya está establecido en la tabla.





# ¿Y SI CONCATENAMOS COLUMNAS?

- Vamos a utilizar la palabra reservada **CONCAT** para mostrar dos o más datos en una misma columna.
- Pero... **¿Qué pasará con el nombre de esa columna?**
- No quedará muy presentable, y es en ese momento donde juega un papel aún más importante nuestra palabra reservada AS para otorgarle un Alias a esta nueva columna.
- **SELECT CONTACT(columna1, columna2, ...)**  
*AS alias FROM nombre\_tabla;*





# AGRUPACIÓN DE DATOS



# AGRUPAR DATOS

- Por medio de **GROUP BY** tenemos la capacidad de agrupar datos.
- Su verdadero potencial se saca en consultas complejas, pero de lo más básico que podemos obtener es la cantidad de veces que se repite un dato, y agruparlo en diferentes columnas.
- Esto último se puede lograr en conjunto con COUNT().
- `SELECT nombre_columna,  
COUNT(nombre_columna) FROM  
nombre_tabla GROUP BY nombre_columna;`



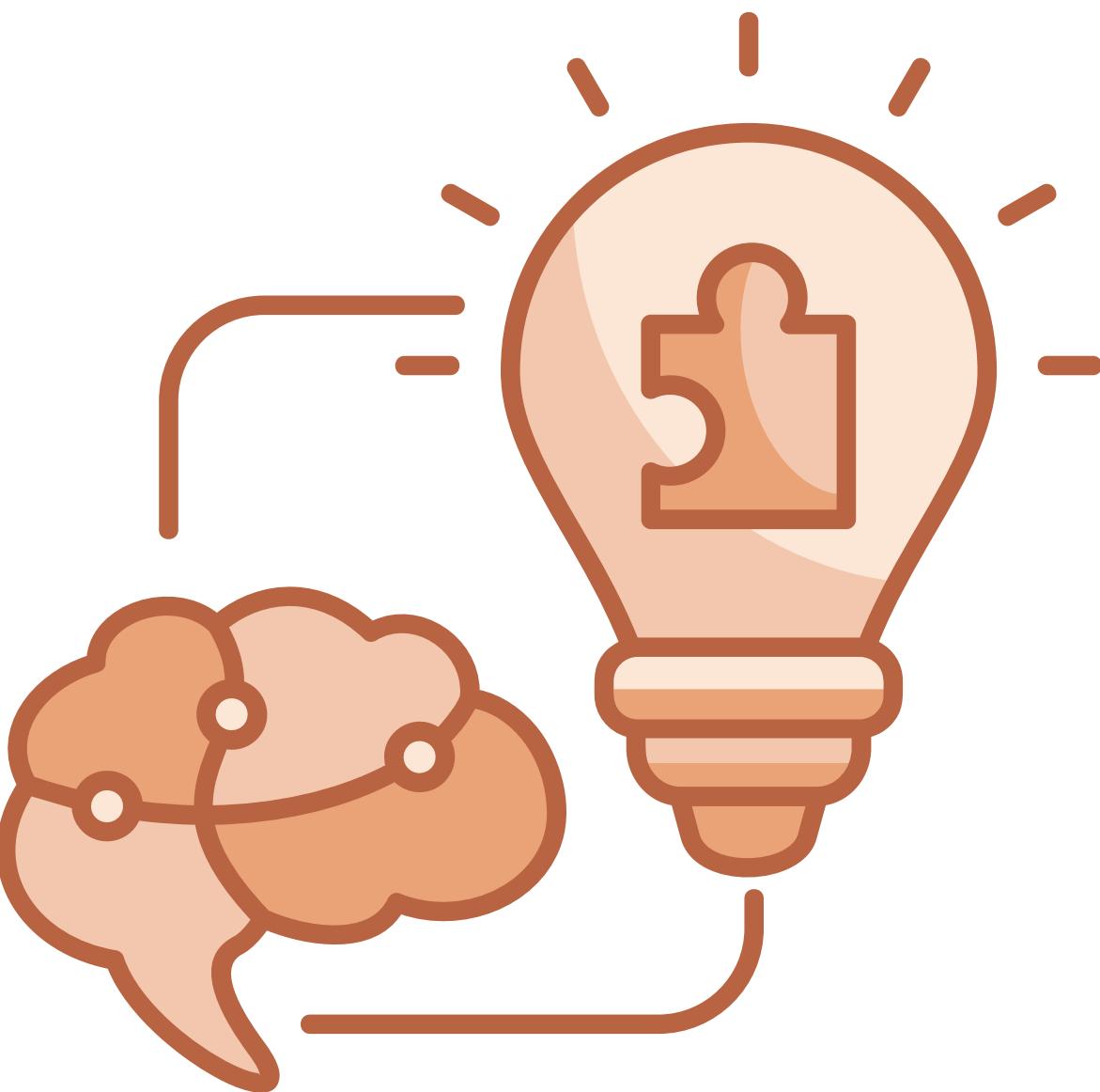


# ¿ES POSIBLE APLICAR LÓGICA DENTRO DE SQL?



# ¿ESTRUCTURA DE CONTROL?

- Existe una manera de aplicar cierta lógica dentro de SQL que nos va a permitir crear una nueva columna a partir de la manipulación de los datos que ya tenemos.
- ```
SELECT nombre_tabla,  
CASE  
    WHEN condición THEN bloque_true  
    ELSE bloque_false  
END AS alias  
FROM nombre_tabla;
```





**GRACIAS POR  
LA ATENCIÓN**