

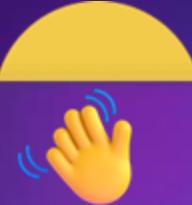
Clase 6. Bootcamp de SQL profesional

# Tipos de Datos y Constraints

Keyla Dolores



# Hola, soy Keyla Dolores



## PUNTOS DE CONTACTO

LinkedIn: [Keyla Dolores Méndez](#)

Blog: [www.keyladolores.com](#)

Twitter: @KeylaDoloresM



## EXPERIENCIA

Data Architect

Máster en Big Data & Analytics

Speaker Azure y Big Data

Blogger

Microsoft Certified Trainer

## CERTIFICACIONES



## CONTENIDO



# Contenido de la Clase

- /01 Tipos de datos en MySQL
- /02 Funciones para manipular datos
- /03 Constraints
- /04 Desafíos finales
- /05 Preguntas

# Empecemos...



# ¿Por qué es importante elegir el tipo de dato correcto?

- Un mal tipo de dato puede hacer que una base de datos sea ineficiente y consuma más recursos de lo necesario.
- La elección adecuada optimiza el rendimiento y la integridad de los datos.
- Veremos cómo tomar mejores decisiones al definir estructuras en SQL.

# ¿Qué tipos de datos existen en MySQL?

- Cadenas de texto: CHAR, VARCHAR, TEXT
- Números: INT, DECIMAL, FLOAT, DOUBLE
- Fechas y horas: DATE, DATETIME, TIMESTAMP
- Booleanos: BOOLEAN (TINYINT(1))



# Diferencia entre VARCHAR y TEXT

- VARCHAR(n): Adecuado para textos cortos con indexación eficiente.
- TEXT: Permite almacenar grandes volúmenes de texto, pero sin índices eficientes.

¡Vamos a la demo!

📌 Insertar datos en ambas columnas

# Diferencia entre INT vs DECIMAL

- INT: Solo almacena valores enteros.
- DECIMAL(m,d): Permite valores con precisión decimal, ideal para montos de dinero.

¡Reto!

📌 Agregar precio\_descuento con DECIMAL(10,2)

# Diferencia entre DATETIME vs TIMESTAMP

- DATETIME: No cambia con la zona horaria.
- TIMESTAMP: Se ajusta automáticamente según la zona horaria del servidor.

¡Vamos a la demo!

📌 Ver datos en clientes.fecha\_registro

# ¿Cómo transformar datos en SQL?

- CAST(columna AS CHAR): Convierte valores numéricos a texto.
- ROUND(valor, 2): Redondea valores decimales.

¡Vamos a la demo!

📌 Convertir precios en texto y redondear valores.

# ¿Qué son las constraints y por qué son importantes?

- Las constraints garantizan la integridad de los datos.
- Principales constraints en MySQL:
  - ✓ NOT NULL: Evita valores nulos.
  - ✓ DEFAULT: Asigna un valor por defecto.
  - ✓ CHECK: Restringe valores posibles en una columna.
  - ✓ UNIQUE: Garantiza que los valores sean únicos.

## Aplica lo aprendido en la base de datos

- 📌 La empresa detectó inconsistencias en la BD.
  - 📌 Objetivo: Corregir estructura aplicando constraints adecuadas.
- 
- 📌 Ejercicios:
  - ✓ Hacer que email sea obligatorio y único en clientes.
  - ✓ Evitar precios negativos en productos.
  - ✓ Asegurar que pedidos.cantidad sea al menos 1.
  - ✓ Crear estado\_pedido con valores válidos.
  - ✓ Evitar stock negativo.

# Desafío Adicional para Expertos

- 📌 Objetivo: Refinar aún más la BD.
- 📌 Ejercicios:
  - ✓ Garantizar que `productos.nombre` tenga al menos 3 caracteres.
  - ✓ Impedir que `productos.stock` supere las 1000 unidades.
  - ✓ Agregar una **constraint CHECK** en `productos.precio` para que no sea menor a 0.
  - ✓ Modificar `clientes.email` para que **no permita valores nulos (NOT NULL)**.
  - ✓ Agregar una restricción **UNIQUE** en `productos.nombre`.

# Nivel Extremo: "Registro de Facturas"

## 📌 Escenario:

Tienes que diseñar la estructura de una tabla facturas para una empresa, pero con **estas reglas específicas**:

## 📌 Requisitos obligatorios:

- 1** Cada factura debe tener un número único (`factura_id`).
- 2** Se debe registrar la **fecha de emisión**.
- 3** El **monto total** no puede ser negativo.
- 4** Solo puede haber **tres estados posibles** ("pendiente", "pagada", "anulada").
- 5** La factura debe estar asociada a un cliente (`cliente_id`).
- 6** Si no se especifica el estado, debe asumir "pendiente" por defecto.
- 7** Solo se pueden registrar facturas desde el **año 2000 en adelante**.

# Nivel Extremo: "Registro de Facturas"

## 📌 Escenario:

Tienes que diseñar la estructura de una tabla facturas para una empresa, pero con **estas reglas específicas**:

## 📌 Requisitos obligatorios:

- 1** Cada factura debe tener un número único (`factura_id`).
- 2** Se debe registrar la **fecha de emisión**.
- 3** El **monto total** no puede ser negativo.
- 4** Solo puede haber **tres estados posibles** ("pendiente", "pagada", "anulada").
- 5** La factura debe estar asociada a un cliente (`cliente_id`).
- 6** Si no se especifica el estado, debe asumir "pendiente" por defecto.
- 7** Solo se pueden registrar facturas desde el **año 2000 en adelante**.

# ¿Preguntas?

