Nombre de la práctica	Restricciones			No.	2
Asignatura:	Taller de bases de datos	Carrera:	ISIC	Duración de la práctica (Hrs)	

I. Nombre: Jocelin Reyes Rodriguez

II. Lugar de realización de la práctica (laboratorio, taller, aula u otro): Aula

III. Material empleado:
Computadora

IV. Desarrollo de la práctica:
Problemario DDL y Constraints en MySQL.

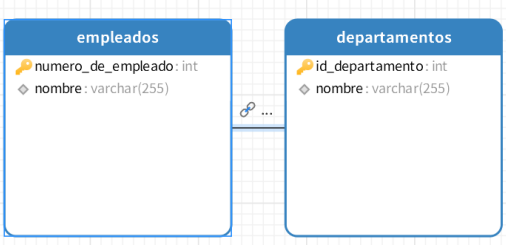
Problema 1: Registro de empleados con restricciones salariales
Una empresa quiere guardar los Empleados, no cuentan con un registro como numero de empleados, anteriormente se tenia registro en un Excel con el nombre, email y salario, las reglas de la empresa no permiten un salario mensual menor a $3000 ni mayor a $50000. Crea la base de datos llamada Ejercicio_Constraints_1 y la tabla Empleados que cumpla con estos requisitos.

```
CREATE TABLE Empleados(
id_empledo INT PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(30) NOT NULL,
ap1 VARCHAR(30) NOT NULL,
ap2 VARCHAR(30) NOT NULL,
email VARCHAR(40) NOT NULL UNIQUE,
salario INT,
CHECK(salario >= 3000 & salario <= 50000)
);
```

Problema 2: Relación entre empleados y departamentos
Una empresa necesita organizar a sus empleados según los departamentos a los que pertenecen. Cada departamento ene un nombre único. La empresa quiere almacenar la información de los empleados junto con el departamento al que están asignados. Actualmente, cada empleado tiene un número de identificación Numero de empleado y el nombre del departamento. Se requiere crear una relación entre ambas tablas para que cada empleado esté asignado a un departamento. Crea la base de datos llamada Ejercicio_Constraints_2 y las tablas Empleados y Departamentos con las restricciones necesarias para cumplir con esta relación.

```
CREATE TABLE Empleados (
numero_de_empleado INT PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(255),
FOREIGN KEY (nombre) REFERENCES departamentos(nombre)
);

CREATE TABLE Departamentos (
id_departamento INT PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(255) UNIQUE NOT NULL
);
```



Problema 3: Control de inventario de productos

Una enda quiere llevar el control de los productos que vende. La información almacenada en la tabla de Productos debe incluir el nombre del producto, código de barras y su precio. A su vez, necesitan añadir una columna para el stock de cada producto, la cual no puede ser nula y debe tener un valor por defecto de 100. Además, el precio de los productos debe ser siempre mayor a 0, y el nombre de cada producto debe ser único para evitar duplicados.

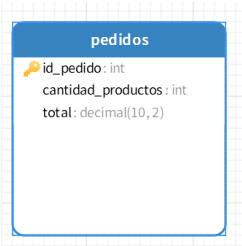
Crea la base de datos llamada Ejercicio_Constraints_3 y realiza las modificaciones necesarias en la tabla Productos para cumplir con estos requisitos.

```
CREATE TABLE Productos (
    id_producto INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(255) UNIQUE NOT NULL,
    codigo_barras VARCHAR(255) NOT NULL,
    precio DECIMAL(10, 2) CHECK (precio > 0) NOT NULL,
    stock INT DEFAULT 100 NOT NULL
);
```

Problema 4: Control de pedidos con validación de montos

Una empresa quiere registrar los pedidos que recibe, pero necesita asegurarse de que el total de cada pedido sea proporcional a la can dad de productos solicitados. Específicamente, el total de cada pedido debe ser al menos igual a la can dad de productos mul plicada por 10. Además, cada pedido debe contener al menos 1 producto. Crea la base de datos llamada Ejercicio_Constraints_4 y la tabla Pedidos que cumpla con estas validaciones usando restricciones CHECK.

```
CREATE TABLE Pedidos (
    id_pedido INT AUTO_INCREMENT PRIMARY KEY,
    cantidad_productos INT NOT NULL CHECK (cantidad_productos >= 1),
    total DECIMAL(10, 2) NOT NULL,
    CHECK (total >= cantidad_productos * 10)
);
```



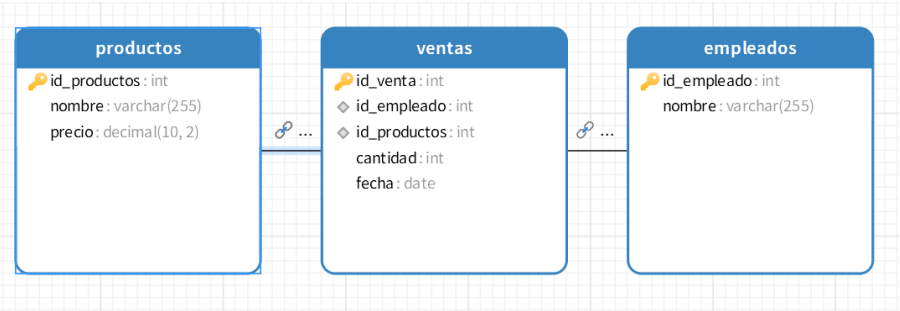
Problema 5: Control de ventas de productos por empleados

Una empresa de ventas necesita registrar las ventas que realiza. Cada venta está asociada a un empleado y a un producto específico. Para garantizar la integridad de los datos, se requiere que cada venta tenga la referencia tanto del empleado como del producto, y que las ventas sean realizadas en una fecha válida (no futura). Además, la can dad de productos vendidos debe ser mayor a 0. Crea la base de datos llamada Ejercicio_Constraints_5 y las tablas necesarias para almacenar esta información, incluyendo las claves foráneas para relacionar las tablas de empleados y productos con las ventas.

```
CREATE TABLE Empleados (
    id_empleado INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(255) NOT NULL
);

CREATE TABLE Productos (
    id_productos INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(255) NOT NULL,
    precio DECIMAL(10, 2) NOT NULL CHECK (precio > 0)
);

CREATE TABLE Ventas (
    id_venta INT AUTO_INCREMENT PRIMARY KEY,
    id_empleado INT,
    id_productos INT,
    cantidad INT NOT NULL CHECK (cantidad > 0),
    fecha DATE NOT NULL CHECK (fecha <= "2024-09-26"),
    FOREIGN KEY (id_empleado) REFERENCES empleados(
id_empleado),
    FOREIGN KEY (id_productos) REFERENCES productos(
id_productos)
);
```



V. Conclusiones:

Las restricciones en SQL son reglas que se aplican a las columnas de una base de datos para garantizar la integridad y calidad de los datos. Estas restricciones, como PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK y NOT NULL, ayudan a prevenir la entrada de datos erróneos o inconsistentes, asegurando que los registros sean válidos y cumplan con las reglas de negocio establecidas. Al implementar restricciones, se fomenta un entorno de datos más seguro y confiable, facilitando la gestión y consulta de la información a lo largo del tiempo. Sin embargo, es importante balancear el uso de restricciones para no limitar la flexibilidad en el manejo de los datos.