Nombre de la práctica	Uso de Constrains			No.	01
Asignatura:	Taller de bases de datos	Carrera:	Ingeniería en sistemas	Duración de la práctica (Hrs)	

Nombre del alumno: Jesús Navarrete Martínez
Grupo: 3501

I. Competencia(s) específica(s):

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

III. Material empleado:

- **Equipo de computo**

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 número de empleados, anteriormente se tenía 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.

Creación de tablas y definición de atributos para la base de datos mediante lenguaje DDL.

```
CREATE TABLE empleados (
id_empleado INT PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(30) NOT NULL,
apellido1 VARCHAR (30) NOT NULL,
apellido2 VARCHAR (30) NOT NULL,
email VARCHAR(100) NOT NULL CHECK (email LIKE '%_@__%.__%'),
sueldo INT NOT NULL CHECK (sueldo>=3000 & sueldo<=50000)
);
```

Tablas Resultantes:

Problema 2: Relación entre empleados y departamentos

Una empresa necesita organizar a sus empleados según los departamentos a los que pertenecen. Cada departamento tiene 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.

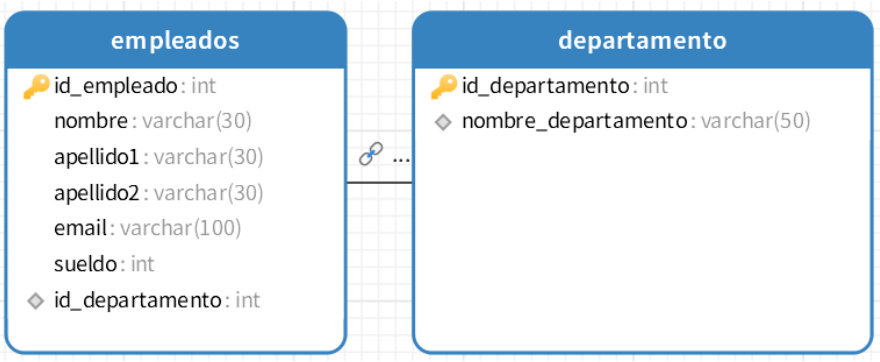
Crea la base de datos llamada Ejercicio_Constraints_2 y las tablas Empleados y Departamentos con las restricciones necesarias para cumplir con esta relación.

Creación de tablas y definición de atributos para la base de datos mediante lenguaje DDL.

```
CREATE TABLE departamento (
id_departamento INT PRIMARY KEY AUTO_INCREMENT,
nombre_departamento VARCHAR(50) NOT NULL UNIQUE
);

CREATE TABLE empleados (
id_empleado INT PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(30) NOT NULL,
apellido1 VARCHAR (30) NOT NULL,
apellido2 VARCHAR (30) NOT NULL,
email VARCHAR(100) NOT NULL CHECK (email LIKE '%_@__%.__%'),
sueldo INT NOT NULL CHECK (sueldo>=3000 & sueldo<=50000),
id_departamento INT,
FOREIGN KEY ( id_departamento) REFERENCES departamento(id_departamento) ON UPDATE CASCADE ON DELETE RESTRICT
);
```

Tablas Resultantes:



Problema 3: Control de inventario de productos

Una tienda 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.

Creación de tablas y definición de atributos para la base de datos mediante lenguaje DDL.

```
CREATE TABLE productos (
id_producto INT PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(30) NOT NULL UNIQUE,
codigo_barras VARCHAR(100) NOT NULL UNIQUE,
precio INT NOT NULL CHECK (precio>0),
stock  INT NOT NULL DEFAULT 100
);

```

Tablas Resultantes:

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 cantidad de productos solicitados. Específicamente, el total de cada pedido debe ser al menos igual a la cantidad de productos multiplicada 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 .

Creación de tablas y definición de atributos para la base de datos mediante lenguaje DDL.

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

```

Tablas Resultantes:

Problema 5: Control de ventas de productos por empleados

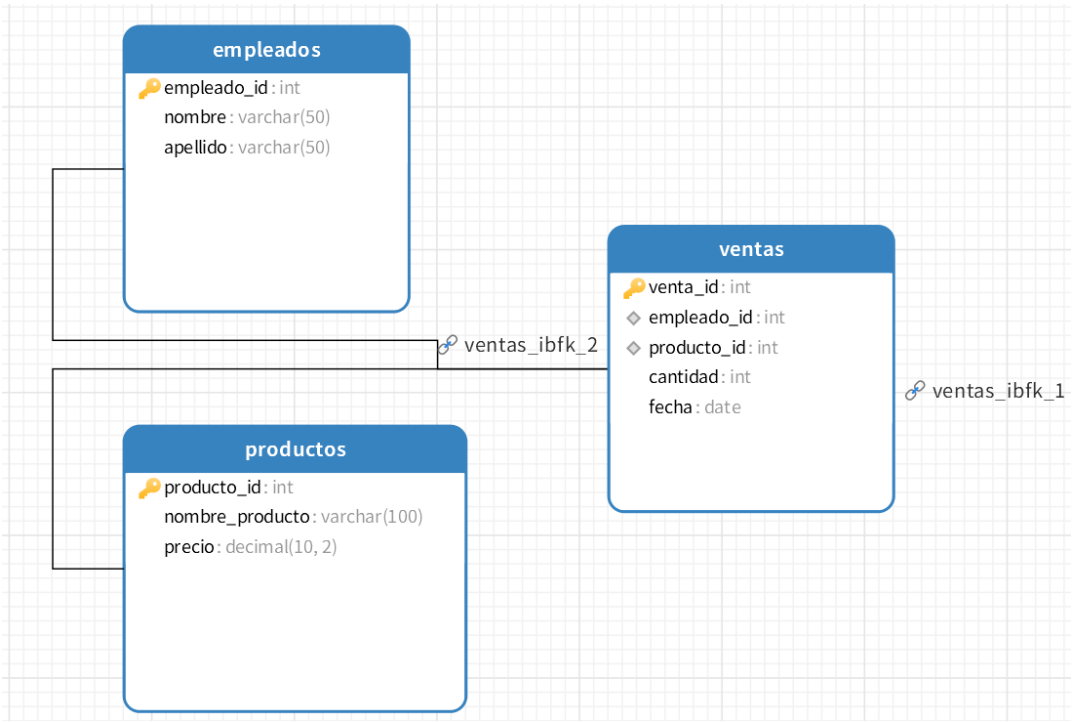
Una empresa de ventas necesita registrar las ventas que realiza. Cada venta está asociada a un empleado y a un producto específico. Para garantizar la integridad de los datos, se requiere que cada venta tenga la referencia tanto del empleado como del producto, y que las ventas sean realizadas en una fecha válida (no futura). Además, la cantidad 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.

Creación de tablas y definición de atributos para la base de datos mediante lenguaje DDL.

```
1
2  CREATE TABLE empleados (
3      empleado_id INT AUTO_INCREMENT PRIMARY KEY,
4      nombre VARCHAR(50),
5      apellido VARCHAR(50)
6  );
7
8
9  CREATE TABLE productos (
10     producto_id INT AUTO_INCREMENT PRIMARY KEY,
11     nombre_producto VARCHAR(100),
12     precio DECIMAL(10,2)
13 );
14
15 -- Tabla de ventas
16 CREATE TABLE ventas (
17     venta_id INT AUTO_INCREMENT PRIMARY KEY,
18     empleado_id INT,
19     producto_id INT,
20     cantidad INT CHECK (cantidad > 0),
21     fecha DATE DEFAULT (CURRENT_DATE),
22     FOREIGN KEY (empleado_id) REFERENCES empleados(empleado_id),
23     FOREIGN KEY (producto_id) REFERENCES productos(producto_id)
24 );
```

Tablas Resultantes:



V. Conclusiones:

La correcta definición de una base de datos utilizando **DDL** (Lenguaje de Definición de Datos) y la implementación adecuada de **restricciones** (**constraints**) es fundamental para garantizar la **integridad** y **consistencia** de los datos. A lo largo de esta práctica, se ha demostrado cómo las distintas restricciones, como las claves primarias (**PRIMARY KEY**), claves foráneas (**FOREIGN KEY**), y las restricciones de unicidad (**UNIQUE**) o validación de condiciones (**CHECK**), permiten establecer reglas que protegen la validez de los datos insertados en las tablas. Además, se evidenció la importancia de gestionar correctamente las relaciones entre tablas mediante **claves foráneas**, asegurando que las referencias entre entidades se mantengan coherentes. También se analizaron las limitaciones en MySQL, como la imposibilidad de usar funciones en restricciones CHECK, lo que plantea la necesidad de soluciones alternativas, como el uso de **triggers** o la implementación de validaciones a nivel de aplicación.