Nombre de la práctica	CONSTRAINTS			No.	3
Asignatura:	Taller de bases de datos	Carrera:	Ingeniería de sistemas computacionales	Duración de la práctica (Hrs)	2hrs

Alumno: Francisco David Colin Lira

I. Competencia(s) específica(s):
Construye expresiones en SQL para resolver necesidades de recuperación de información con las reglas sintácticas del lenguaje de manipulación de datos.

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

III. Material empleado:
Computadora y gestor de base de datos

IV. Desarrollo de la práctica:

Ejercicio 1:
Registro de empleados con 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 DATABASE Ejercicio_Constraints_1;
USE Ejercicio_Constraints_1;

CREATE TABLE Empleados (
    No_Empleado INT AUTO_INCREMENT,
    Nombre_Empleado VARCHAR(30) NOT NULL,
    Apellido1_Empleado VARCHAR(30) NOT NULL,
    Apellido2_Empleado VARCHAR(30),
    Email_Empleado VARCHAR(30) NOT NULL UNIQUE,
    Salario INT NOT NULL CHECK (Salario >= 3000 AND Salario <= 50000),
    PRIMARY KEY (No_Empleado)
);
```

Ejercicio 2:
Relación entre empleados y departamentos:

Una empresa necesita organizar a sus empleados según los departamentos a los que pertenecen. Cada departamento información de los empleados junto con el departamento al que están asignados. Actualmente, cada empleado 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 DATABASE Ejercicio_Constraints_2;
USE Ejercicio_Constraints_2;


CREATE TABLE Departamento(
    Nombre_Departamento VARCHAR(50) UNIQUE NOT NULL,
    PRIMARY KEY (Nombre_Departamento)
);

## EMPLEADO --> DEPARTAMENTOS 1,1
## DEPARTAMENTO --> EMPLEADOS  1,n

## RELACION GENERAL 1:N

CREATE TABLE Empleado(
    Numero_Empleado INT AUTO_INCREMENT,
    Nombre_Empleado VARCHAR(30) NOT NULL,
    Apellido1_Empleado VARCHAR(30) NOT NULL,
    Apellido2_Empleado VARCHAR(30),
    Nombre_Departamento VARCHAR(50),
    PRIMARY KEY (Numero_Empleado),
    FOREIGN KEY (Nombre_Departamento) REFERENCES Departamento(Nombre_Departamento)
);
```

Ejercicio 3:
Control de inventario de productos:

Una 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 DATABASE Ejercicio_Constraints_3;
USE Ejercicio_Constraints_3;

CREATE TABLE Producto(
    Codigo_Barras INT NOT NULL,
    Nombre_Producto VARCHAR(40) NOT NULL UNIQUE,
    Precio_Producto DECIMAL(10, 2) NOT NULL CHECK (Precio_Producto > 0),
    Stock INT DEFAULT 100 NOT NULL,
    PRIMARY KEY (Codigo_Barras)
);
```

Ejercicio 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 debe ser al menos igual a la cantidad de productos multiplicada por 10. Ademas, cada pedido debe tener amenos 1 producto.

Crea la base de datos llamada Ejercicio_Constraints_4 y la tabla Pedidos que cumpla con estas validaciones usando restricciones CHECK.

```
CREATE DATABASE Ejercicio_Constraints_4;
USE Ejercicio_Constraints_4;

CREATE TABLE Pedidos (
    Id_Pedido INT AUTO_INCREMENT,
    Cantidad_Productos INT NOT NULL DEFAULT 1 CHECK (Cantidad_Productos >= 1),
    Total INT NOT NULL,
    CHECK (Total >= Cantidad_Productos * 10),
    PRIMARY KEY (Id_Pedido)
);
```

Ejercicio 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.

```sql
CREATE DATABASE Ejercicio_Constraints_5;
USE Ejercicio_Constraints_5;

CREATE TABLE Empleado(
    Id_empleado INT PRIMARY KEY AUTO_INCREMENT,
    Nombre_empleado VARCHAR(30) NOT NULL,
    Apellido1_empleado VARCHAR (30) NOT NULL,
    Apellido2_empleado VARCHAR(30)
);

CREATE TABLE Producto(
    Id_producto INT PRIMARY KEY AUTO_INCREMENT,
    Nombre_producto VARCHAR(30) NOT NULL,
    Precio_producto DECIMAL(10, 2) NOT NULL
);

CREATE TABLE Venta(
    Id_empleado INT,
    Id_producto INT,
    Fecha_venta DATE NOT NULL,
    Cantidad_productos INT NOT NULL,



    CHECK (Cantidad_productos > 0),

    FOREIGN KEY (Id_empleado) REFERENCES Empleado(Id_empleado),
    FOREIGN KEY (Id_producto) REFERENCES Producto(Id_producto),
    PRIMARY KEY (Id_empleado, Id_producto)
);

-- En mi caso no tengo como tal MySQL y no me funciona esta funcion de CURRENT_DATE o CURDATE()
-- Pero asi es como se hace
ALTER TABLE Venta ADD CHECK (Fecha_venta <= CURRENT_DATE);
```

V. Conclusiones:

Pues me pareció bastante practico el echo de usar estas restricciones para asi los datos tengan o se mantengan un poco mas íntegros y algunos que no sabia que lo eran pero son restricciones como tal nativas por asi decirlo ya que son muy comunes y pues otras que no sabia tanto o no sabia usado eran los CHECK pero con estas prácticas ya me quedo bastante mas claro la verdad y pues me gusto hacerla solo que en el ultimo ejercicio tuve pequeños detalles pero investigándomela me di cuenta que es mi versionada de MySQL ya que no como tal no lo es pero es un derivado y por eso es que no cuenta con esta función para extraer la fecha actual.