UNIVERSIDAD DE COLIMA

Facultad de Telemática

Bases de Datos Avanzadas

A5 IMPLEMENTACIÓN DE FUNCIONES, PROCEDIMIENTOS ALMACENADOS Y DISPARADORES

Maestro: GALLARDO HEREDIA ARMANDO ROMAN

Alumno: VIZCAINO LUPIAN ALEJANDRO

#PAPELERÍA

```
DROP DATABASE IF EXISTS papeleria;
CREATE DATABASE papeleria;
USE papeleria;
```

#Tablas

```
CREATE TABLE Categorias (
Id_Categoria INT AUTO_INCREMENT PRIMARY KEY,
Nombre VARCHAR(100) NOT NULL
);

CREATE TABLE Proveedores (
Id_Proveedor INT AUTO_INCREMENT PRIMARY KEY,
Nombre VARCHAR(100) NOT NULL,
Contacto VARCHAR(100) NOT NULL
);

CREATE TABLE Productos (
Id_Producto INT AUTO_INCREMENT PRIMARY KEY,
Nombre VARCHAR(100) NOT NULL,
Descripcion TEXT,
Precio DECIMAL(10, 2) NOT NULL,
Stock INT NOT NULL,
Id_Categoria INT NOT NULL,
Id_Proveedor INT NOT NULL,
FOREIGN KEY (Id_Categoria) REFERENCES Categorias(Id_Categoria),
FOREIGN KEY (Id_Proveedor) REFERENCES Proveedores(Id_Proveedor)
);


CREATE TABLE Clientes (
Id_Cliente INT AUTO_INCREMENT PRIMARY KEY,
Nombre VARCHAR(100) NOT NULL,
ApellIdo VARCHAR(100) NOT NULL,
Correo VARCHAR(100) NOT NULL UNIQUE,
Telefono VARCHAR(20),
Direccion VARCHAR(100)
);

CREATE TABLE Ventas (
```

```
Id_Venta INT AUTO_INCREMENT PRIMARY KEY,
Id_Cliente INT,
Fecha TIMESTAMP,
Total DECIMAL(10, 2) NOT NULL,
Metodo_Pago VARCHAR(50),
FOREIGN KEY (Id_Cliente) REFERENCES Clientes(Id_Cliente)
);

CREATE TABLE Detalle_de_Venta (
Id_Detalle INT AUTO_INCREMENT PRIMARY KEY,
Id_Venta INT,
Id_Producto INT,
Cantidad INT NOT NULL,
Subtotal DECIMAL(10, 2) NOT NULL,
FOREIGN KEY (Id_Venta) REFERENCES Ventas(Id_Venta),
FOREIGN KEY (Id_Producto) REFERENCES Productos(Id_Producto)
);
```

#Inserts

 # INSERTAR CATEGORÍAS en la tabla de Categorías:

```
INSERT INTO Categorias (Nombre) VALUES ('Oficina');
INSERT INTO Categorias (Nombre) VALUES ('Papelería');
INSERT INTO Categorias (Nombre) VALUES ('Escolar');
```

 # INSERTAR PROVEEDORES en la tabla de Proveedores:

```
INSERT INTO Proveedores (Nombre, Contacto) VALUES ('Papelería La Normal',
'laniormal@papeleria.com');
INSERT INTO Proveedores (Nombre, Contacto) VALUES ('Distribuldora Escolar',
'contacto@distribuldoraescolar.com');
INSERT INTO Proveedores (Nombre, Contacto) VALUES ('Oficina Oracle',
'ventas@oficinaoracle.com');
```

 # INSERTAR PRODUCTOS en la tabla de Productos:

```
INSERT INTO Productos (Nombre, Descripcion, Precio, Stock, Id_Categoria,
id_proveedor) VALUES ('Libreta norma', 'Cuaderno de 100 hojas, tamaño A4',
45.50, 150,3,2);

INSERT INTO Productos (Nombre, Descripcion, Precio, Stock, id_categoria,
id_proveedor) VALUES ('Bolígrafo Azul', 'Bolígrafo tinta azul, punta fina', 12.00,
300,1,1);
```

INSERT INTO Productos (Nombre, Descripcion, Precio, Stock, id_categoria, id_proveedor) VALUES ('Tijeras', 'Tijeras de acero inoxIdable', 25.00, 80,2,1);

 # INSERTAR CLIENTES en la tabla de Clientes:
INSERT INTO Clientes (Nombre, ApellIdo, Correo, Telefono, Direccion) VALUES ('Juan', 'Pérez', 'jp@correo.com', '555-1234', 'Calle Falsa 123');

INSERT INTO Clientes (Nombre, ApellIdo, Correo, Telefono, Direccion) VALUES ('María', 'García', 'mg@correo.com', '555-5678', 'AvenIda Siempreviva 742');

INSERT INTO Clientes (Nombre, ApellIdo, Correo, Telefono, Direccion) VALUES ('Carlos', 'López', 'cl@correo.com', '555-8765', 'Boulevard de los Sueños 456');

#INSERTAR REGISTROS en la tabla de Ventas:
INSERT INTO Ventas (Id_Cliente, Fecha, Total, Metodo_Pago) VALUES (1, CURRENT_TIMESTAMP, 57.50, 'Tarjeta de Crédito');

INSERT INTO Ventas (Id_Cliente, Fecha, Total, Metodo_Pago) VALUES (2, CURRENT_TIMESTAMP, 90.00, 'Efectivo');

INSERT INTO Ventas (Id_Cliente, Fecha, Total, Metodo_Pago) VALUES (3, CURRENT_TIMESTAMP, 25.00, 'Transferencia Bancaria');

#INSERTAR REGISTROS en la tabla Detalle_de_Venta:
INSERT INTO Detalle_de_Venta (Id_Venta, Id_Producto, Cantidad, Subtotal) VALUES (1, 1, 1, 45.50);

INSERT INTO Detalle_de_Venta (Id_Venta, Id_Producto, Cantidad, Subtotal) VALUES (2, 2, 5, 60.00);

INSERT INTO Detalle_de_Venta (Id_Venta, Id_Producto, Cantidad, Subtotal) VALUES (3, 3, 1, 25.00);

#Views

 # 1.-Esta vista muestra todos los productos junto con su categoría correspondiente.

CREATE VIEW/ categoria_producto AS
SELECT
productos.nombre AS Nombre_Producto,

```
categorias.nombre AS Nombre_Categoria
FROM
productos
JOIN
categorias ON productos.id_categoria = categorias.id_categoria;
```

#2.-Esta vista muestra los productos que tienen un stock de 0.

```
CREATE VIEW Productos_Sin_Stock AS
SELECT Id_Producto, Nombre, Descripcion, Precio
FROM Productos
WHERE Stock = 0;
```

#3.-Esta vista muestra las ventas diarias junto con la fecha correspondiente.

```
CREATE VIEW Ventas_Diarias AS
SELECT DATE(Fecha) AS Fecha, COUNT(Id_Venta) AS Total_Ventas, SUM(Total)
AS Total_Diario
FROM Ventas
GROUP BY DATE(Fecha);
```

#4.-Esta vista muestra los proveedores junto con los productos que suministran.

```
CREATE VIEW Proveedores_Productos AS
SELECT Pr.Id_Proveedor, Pr.Nombre AS Nombre_Proveedor, P.Nombre AS
Nombre_Producto
FROM Proveedores Pr
JOIN Productos P ON Pr.Id_Proveedor = P.Id_Proveedor;
```

#5.- Esta vista muestra la cantidad de productos en cada categoría.

```
CREATE VIEW Cantidad_Productos_Categoria AS
SELECT C.Nombre AS Nombre_Categoria, COUNT(P.Id_Producto) AS
Cantidad_Productos
FROM Categorias C
JOIN Productos P ON C.Id_Categoria = P.Id_Categoria
GROUP BY C.Nombre;
```

#6.-Esta vista muestra las ventas mensuales junto con el mes correspondiente.

```
CREATE VIEW Ventas_Mensuales AS
SELECT DATE_FORMAT(Fecha, '%m-%Y') AS Mes, COUNT(Id_Venta) AS
Total_Ventas, SUM(Total) AS Total_Mensual
FROM Ventas
GROUP BY DATE_FORMAT(Fecha, '%m-%Y');
```

#7.-Esta vista muestra los productos junto con su proveedor correspondiente.

```
CREATE VIEW Productos_Proveedores AS
SELECT P.Id_Producto, P.Nombre AS Nombre_Producto, Pr.Nombre AS
Nombre_Proveedor
FROM Productos P
JOIN Proveedores Pr ON P.Id_Producto = Pr.Id_Proveedor;
```

#8.-Esta vista muestra los usuarios que han realizado las compras más grandes.

```
CREATE VIEW Clientes_Mayores_Compras AS
SELECT C.Id_Cliente, C.Nombre, C.ApellIdo, SUM(V.Total) AS Total_Compras
FROM Clientes C
JOIN Ventas V ON C.Id_Cliente = V.Id_Cliente
GROUP BY C.Id_Cliente
ORDER BY Total_Compras DESC;
```

#9.-Esta vista muestra los productos vendidos cada mes.

```
CREATE VIEW Productos_Vendidos_Mensuales AS
SELECT DATE_FORMAT(V.Fecha, '%m-%Y') AS Mes, P.Nombre AS
Nombre_Producto, SUM(DV.Cantidad) AS Cantidad_Vendida
FROM Ventas V
JOIN Detalle_de_Venta DV ON V.Id_Venta = DV.Id_Venta
JOIN Productos P ON DV.Id_Producto = P.Id_Producto
GROUP BY DATE_FORMAT(V.Fecha, '%m-%Y'), P.Nombre;
```

#10.- Esta vista muestra los usuarios junto con el total de sus compras.

```
CREATE VIEW Clientes_Total_Compras AS
SELECT C.Id_Cliente, C.Nombre, C.ApellIdo, SUM(V.Total) AS Total_Compras
FROM Clientes C
JOIN Ventas V ON C.Id_Cliente = V.Id_Cliente
GROUP BY C.Id_Cliente;
```

#Procedimientos almacenados

#Este procedimiento almacenado permitirá registrar una nueva venta, insertará el registro en la tabla "Ventas" y luego también insertará los detalles de los productos vendidos en la tabla Detalle_de_Venta, además de actualizar el stock del producto.

 # Actualizar el inventario de un producto en específico

```
DELIMITER //
```

```
CREATE PROCEDURE ActualizarInventario(
IN p_Id_Producto INT,
IN p_Cantidad INT )
BEGIN
UPDATE Productos
SET Stock = Stock + p_Cantidad
WHERE Id_Producto = p_Id_Producto;
END //
DELIMITER ;

DELIMITER //
CREATE PROCEDURE registrar_venta(
    IN p_id_cliente INT,
    IN p_metodo_pago VARCHAR(50),
    IN p_total DECIMAL(10, 2),
    IN p_id_producto INT,
    IN p_cantidad INT,
    IN p_subtotal DECIMAL(10, 2)
)
BEGIN
    INSERT INTO Ventas (Id_Cliente, Fecha, Total, Metodo_Pago)
    VALUES (p_id_cliente, NOW(), p_total, p_metodo_pago);

    SET @id_venta = LAST_INSERT_ID();

    INSERT INTO Detalle_de_Venta (Id_Venta, Id_Producto, Cantidad, Subtotal)
    VALUES (@id_venta, p_id_producto, p_cantidad, p_subtotal);

    UPDATE Productos
    SET Stock = Stock - p_cantidad
    WHERE Id_Producto = p_id_producto;

END //
DELIMITER ;
```

#Triggers

#Este disparador se activa antes de insertar un nuevo detalle de venta, para asegurarse de que la cantidad solicitada no exceda el stock disponible.

```
DELIMITER //

CREATE TRIGGER validar_stock
BEFORE INSERT ON Detalle_de_Venta
FOR EACH ROW
BEGIN
   DECLARE stock_actual INT;

   SELECT Stock INTO stock_actual FROM Productos WHERE Id_Producto =
NEW.Id_Producto;

   IF stock_actual < NEW.Cantidad THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Stock insuficiente para este producto.';
   END IF;
END //

DELIMITER ;
```

 # Actualizar auto. el stock del producto al registrar un detalle de venta

```
DELIMITER //

CREATE TRIGGER actualizar_stock_despues_venta
AFTER INSERT ON Detalle_de_Venta
FOR EACH ROW
BEGIN
   UPDATE Productos
   SET Stock = Stock - NEW.Cantidad
   WHERE Id_Producto = NEW.Id_Producto;
END //

DELIMITER ;
```

#BLOG VIAJERO

```
DROP DATABASE IF EXISTS blog_viajero;
CREATE DATABASE blog_viajero;
USE blog_viajero;
```

#Tablas

```
CREATE TABLE Usuarios (
Id_Usuario INT AUTO_INCREMENT PRIMARY KEY,
Nombre VARCHAR(100) NOT NULL,
Correo VARCHAR(100) NOT NULL UNIQUE,
Contrasenia VARCHAR(100) NOT NULL,
Fecha_Registro TIMESTAMP
);

CREATE TABLE Articulos (
Id_Articulo INT AUTO_INCREMENT PRIMARY KEY,
Titulo VARCHAR(100) NOT NULL,
ContenIdo TEXT NOT NULL,
Fecha_Publicacion TIMESTAMP,
Id_Usuario INT,
FOREIGN KEY (Id_Usuario) REFERENCES Usuarios(Id_Usuario)
);

CREATE TABLE Comentarios (
Id_Comentario INT AUTO_INCREMENT PRIMARY KEY,
Contenido TEXT,
Fecha TIMESTAMP,
Id_Usuario INT,
Id_Articulo INT,
FOREIGN KEY (Id_Usuario) REFERENCES Usuarios(Id_Usuario),
FOREIGN KEY (Id_Articulo) REFERENCES Articulos(Id_Articulo)
);

CREATE TABLE Etiquetas (
Id_Etiqueta INT AUTO_INCREMENT PRIMARY KEY,
Nombre VARCHAR(100)
);
```

```
CREATE TABLE Relacion_Articulo_Etiqueta (
Id_Articulo_Etiqueta INT AUTO_INCREMENT PRIMARY KEY,
Id_Articulo INT,
Id_Etiqueta INT,
FOREIGN KEY (Id_Articulo) REFERENCES Articulos(Id_Articulo),
FOREIGN KEY (Id_Etiqueta) REFERENCES Etiquetas(Id_Etiqueta)
);
```

#Inserts

#Insertar registros en la tabla Usuarios

```
INSERT INTO Usuarios (Nombre, Correo, Contrasenia, Fecha_Registro)  VALUES
('Ana López', 'ana.lopez@example.com', 'password123', CURRENT_TIMESTAMP),
('Pedro Martínez', 'pedro.martinez@example.com', 'password456',
CURRENT_TIMESTAMP),
('Maria Fernández', 'maria.fernandez@example.com', 'password789',
CURRENT_TIMESTAMP),
('Luis Gómez', 'luis.gomez@example.com', 'password012',
CURRENT_TIMESTAMP),
('Laura García', 'laura.garcia@example.com', 'password345',
CURRENT_TIMESTAMP);
```

#Insertar registros en la tabla Articulos

```
INSERT INTO Articulos (Titulo, Contenido, Fecha_Publicacion, Id_Usuario)
VALUES
('Primer Artículo', 'Contenido del primer artículo.', CURRENT_TIMESTAMP, 1),
('Segundo Artículo', 'Contenido del segundo artículo.', CURRENT_TIMESTAMP, 2),
('Tercer Artículo', 'Contenido del tercer artículo.', CURRENT_TIMESTAMP, 3),
('Cuarto Artículo', 'Contenido del cuarto artículo.', CURRENT_TIMESTAMP, 4),
('Quinto Artículo', 'Contenido del quinto artículo.', CURRENT_TIMESTAMP, 5);
```

#Insertar registros en la tabla Comentarios

```
INSERT INTO Comentarios (Contenido, Fecha, Id_Usuario, Id_Articulo) VALUES
('Comentario sobre el primer artículo.', CURRENT_TIMESTAMP, 1, 1),
('Comentario sobre el segundo artículo.', CURRENT_TIMESTAMP, 2, 2),
('Comentario sobre el tercer artículo.', CURRENT_TIMESTAMP, 3, 3),
('Comentario sobre el cuarto artículo.', CURRENT_TIMESTAMP, 4, 4),
('Comentario sobre el quinto artículo.', CURRENT_TIMESTAMP, 5, 5);
```

#Insertar registros en la tabla Etiquetas
INSERT INTO Etiquetas (Nombre) VALUES
('Tecnología'),
('Ciencia'),
('Viajes'),
('Cultura'),
('Deportes');

#Insertar registros en la tabla Relacion_Articulo_Etiqueta
INSERT INTO Relacion_Articulo_Etiqueta (Id_Articulo, Id_Etiqueta) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5);

#Views

CREATE VIEW vista_usuarios_registrados AS SELECT id_usuario, nombre, correo, fecha_registro FROM usuarios;

CREATE VIEW Vista_Articulos_Populares AS SELECT A.ID_Articulo, A.Titulo, A.Fecha_Publicacion, COUNT(C.ID_Comentario) AS Cantidad_Comentarios FROM Articulos A LEFT JOIN Comentarios C ON A.ID_Articulo = C.ID_Articulo GROUP BY A.ID_Articulo ORDER BY Cantidad_Comentarios DESC;

CREATE VIEW Vista_Comentarios_Recientes AS SELECT C.ID_Comentario, U.Nombre AS Usuario, C.Contenido, C.Fecha FROM Comentarios C JOIN Usuarios U ON C.ID_Usuario = U.ID_Usuario ORDER BY C.Fecha DESC;

CREATE VIEW Vista_Articulos_Por_Autor AS SELECT U.Nombre AS Autor, A.Titulo AS Articulo, A.Fecha_Publicacion FROM Usuarios U JOIN Articulos A ON U.ID_Usuario = A.ID_Usuario;

CREATE VIEW vista_articulos_con_etiquetas AS SELECT a.titulo AS articulo, GROUP_CONCAT(e.nombre SEPARATOR ', ') AS etiquetas FROM articulos a
JOIN relacion_articulo_etiqueta ae ON a.id_articulo = ae.id_articulo
JOIN etiquetas e ON ae.id_etiqueta = e.id_etiqueta GROUP BY a.titulo;

#Procedimientos almacenados

Inserta un artículo y luego asocia una etiqueta con ese artículo.

```
DELIMITER //

CREATE PROCEDURE agregar_articulo(
    IN p_titulo VARCHAR(100),
    IN p_contenido TEXT,
    IN p_id_usuario INT,
    IN p_etiqueta VARCHAR(100)
)
BEGIN
    DECLARE etiqueta_id INT;

    -- Insertar el artículo en la tabla Articulos
    INSERT INTO Articulos (Titulo, Contenido, Fecha_Publicacion, Id_Usuario)
    VALUES (p_titulo, p_contenido, NOW(), p_id_usuario);

    -- Obtener el último Id_Articulo insertado
    SET @id_articulo = LAST_INSERT_ID();

    -- Verificar si la etiqueta ya existe
    SELECT Id_Etiqueta INTO etiqueta_id
    FROM Etiquetas
    WHERE Nombre = p_etiqueta;

    -- Si la etiqueta no existe, crearla
    IF etiqueta_id IS NULL THEN
        INSERT INTO Etiquetas (Nombre) VALUES (p_etiqueta);
        SET etiqueta_id = LAST_INSERT_ID();
    END IF;

    -- Relacionar el artículo con la etiqueta
    INSERT INTO Relacion_Articulo_Etiqueta (Id_Articulo, Id_Etiqueta)
    VALUES (@id_articulo, etiqueta_id);

END //

DELIMITER ;
```

#Triggers

```
#Trigger para evitar comentarios vacíos
DELIMITER $$
CREATE TRIGGER before_insert_comentario
BEFORE INSERT ON Comentarios
FOR EACH ROW
BEGIN
   IF NEW.Contenido = '' THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'El contenido del comentario no puede estar vacío';
   END IF;
   SET NEW.Fecha = CURRENT_TIMESTAMP;
END$$
DELIMITER ;
```

```
#Trigger para borrar relaciones al eliminar un artículo para evitar errores de
llaves foráneas
DELIMITER $$
CREATE TRIGGER borrar_articulo_y_relaciones
BEFORE DELETE ON Articulos
FOR EACH ROW
BEGIN
   DELETE FROM Comentarios WHERE Id_Articulo = OLD.Id_Articulo;
   DELETE FROM relacion_articulo_etiqueta WHERE id_articulo=OLD.Id_Articulo;
END$$
DELIMITER ;
```

#CARTELERA_CINE

```
DROP DATABASE IF EXISTS cartelera_de_peliculas_de_cine;
CREATE DATABASE cartelera_de_peliculas_de_cine;
USE cartelera_de_peliculas_de_cine;
```

#Tablas

```
CREATE TABLE Peliculas (
Id_Pelicula INT AUTO_INCREMENT PRIMARY KEY,
Titulo VARCHAR(100) NOT NULL,
Genero VARCHAR(100),
Director VARCHAR(100),
Sinopsis TEXT,
Duracion INT,
Anio YEAR,
Clasificacion VARCHAR(10)
);

CREATE TABLE Cines (
Id_Cine INT AUTO_INCREMENT PRIMARY KEY,
Nombre VARCHAR(100) NOT NULL,
Direccion VARCHAR(100),
Ciudad VARCHAR(100),
Telefono VARCHAR(20)
);


CREATE TABLE Funciones (
    ID_Funcion INT AUTO_INCREMENT PRIMARY KEY,
    ID_Pelicula INT,
    ID_Cine INT,
    Fecha TIMESTAMP,
    Hora TIME,
    Sala INT,
```

```
    Total_Asientos INT, -- Total de asientos en la sala para esta función
    FOREIGN KEY (ID_Pelicula) REFERENCES Peliculas(ID_Pelicula),
    FOREIGN KEY (ID_Cine) REFERENCES Cines(ID_Cine)
);



CREATE TABLE Boletos (
    ID_Boleto INT AUTO_INCREMENT PRIMARY KEY,
    ID_Funcion INT,
    Cantidad INT, -- Número de boletos vendidos
    Precio DECIMAL(10, 2), -- Precio del boleto
    FOREIGN KEY (ID_Funcion) REFERENCES Funciones(ID_Funcion)
);

CREATE TABLE Ingresos_Cines (
    Id_Cine INT PRIMARY KEY,
    Ingreso_Total DECIMAL(10, 2) DEFAULT 0,
    FOREIGN KEY (ID_Cine) REFERENCES Cines(ID_Cine)
);
```

#Inserts

```
#Insertar registros en la tabla Peliculas
INSERT INTO Peliculas (Titulo, Genero, Director, Sinopsis, Duracion, Anio,
Clasificacion)
VALUES
('Inception', 'Sci-Fi', 'Christopher Nolan', 'A thief who enters the dreams of others to
steal secrets.', 148, 2010, 'PG-13'),
('The Matrix', 'Sci-Fi', 'The Wachowskis', 'A computer hacker learns about the true
nature of reality.', 136, 1999, 'R'),
('Parasite', 'Thriller', 'Bong Joon-ho', 'A poor family schemes to become employed
by a wealthy family.', 132, 2019, 'R'),
('The Shawshank Redemption', 'Drama', 'Frank Darabont', 'Two imprisoned men
bond over a number of years.', 142, 1994, 'R'),
```

('The Godfather', 'Crime', 'Francis Ford Coppola', 'The aging patriarch of an organized crime dynasty transfers control.', 175, 1972, 'R'),
('Pulp Fiction', 'Crime', 'Quentin Tarantino', 'The lives of two mob hitmen, a boxer, and a pair of diner bandits intertwine.', 154, 1994, 'R'),
('The Dark Knight', 'Action', 'Christopher Nolan', 'Batman must accept one of the greatest psychological and physical tests.', 152, 2008, 'PG-13'),
('Forrest Gump', 'Drama', 'Robert Zemeckis', 'The presIdencies of Kennedy and Johnson, the events of Vietnam.', 142, 1994, 'PG-13'),
('Gladiator', 'Action', 'RIdley Scott', 'A former Roman General sets out to exact vengeance against the corrupt emperor.', 155, 2000, 'R'),
('Titanic', 'Romance', 'James Cameron', 'A seventeen-year-old aristocrat falls in love with a kind but poor artist.', 195, 1997, 'PG-13');

#Insertar registros en la tabla Cines
INSERT INTO Cines (Nombre, Direccion, Ciudad, Telefono)
VALUES
('Cinepolis Centro', 'Av. Central 123', 'Ciudad de México', '555-1234'),
('Cinemex Reforma', 'Paseo de la Reforma 456', 'Ciudad de México', '555-5678'),
('Cinepolis Santa Fe', 'Av. Santa Fe 789', 'Ciudad de México', '555-8765'),
('Cinemex Interlomas', 'Boulevard Interlomas 101', 'Ciudad de México', '555-4321'),
('Cinepolis Perisur', 'Av. Perisur 202', 'Ciudad de México', '555-6543');

#Insertar registros en la tabla Funciones
INSERT INTO Funciones (Id_Cine, Id_Pelicula, Fecha, Hora, Sala)
VALUES
(1, 1, '2024-08-22', '19:00:00', 1),
(2, 2, '2024-08-22', '20:00:00', 2),
(3, 3, '2024-08-22', '21:00:00', 3),
(4, 4, '2024-08-22', '22:00:00', 4),
(5, 5, '2024-08-22', '18:00:00', 5),
(1, 6, '2024-08-22', '17:00:00', 1),
(2, 7, '2024-08-22', '16:00:00', 2),
(3, 8, '2024-08-22', '15:00:00', 3),
(4, 9, '2024-08-22', '14:00:00', 4),
(5, 10, '2024-08-22', '13:00:00', 5);

```
#Insertar registros en la tabla Boletos
INSERT INTO Boletos (Id_Funcion, Cantidad, Precio) VALUES (1, 10, 100.00);
INSERT INTO Boletos (Id_Funcion, Cantidad, Precio) VALUES (2, 5, 80.00);
INSERT INTO Boletos (ID_Funcion, Cantidad, Precio) VALUES (3, 5, 100.00);
```

#Views

```
CREATE VIEW Vista_Funciones_Cine AS
SELECT
    Funciones.ID_Funcion,
    Cines.Nombre AS Cine,
    Peliculas.Titulo AS Pelicula,
    Funciones.Fecha,
    Funciones.Hora,
    Funciones.Sala
FROM
    Funciones
JOIN
    Cines ON Funciones.ID_Cine = Cines.ID_Cine
JOIN
    Peliculas ON Funciones.ID_Pelicula = Peliculas.ID_Pelicula
WHERE
    Cines.Nombre = 'Cine Central';
```

#Procedimientos almacenados

#Función para reservar una función (verificar disponibilidad y descontar asientos)

```
DELIMITER $$

CREATE PROCEDURE ReservarFuncion(
    IN id_funcion INT,
```

```
    IN cantidad_boletos INT
)
BEGIN
    DECLARE asientos_disponibles INT;

    SELECT (f.Total_Asientos - COALESCE(SUM(b.Cantidad), 0)) INTO
asientos_disponibles
    FROM Funciones f
    LEFT JOIN Boletos b ON f.ID_Funcion = b.ID_Funcion
    WHERE f.ID_Funcion = id_funcion
    GROUP BY f.ID_Funcion;

    IF asientos_disponibles >= cantidad_boletos THEN
        INSERT INTO Boletos (ID_Funcion, Cantidad)
        VALUES (id_funcion, cantidad_boletos);
    ELSE
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No hay suficientes
asientos disponibles';
    END IF;

END$$

DELIMITER ;
```

#Triggers

```
DELIMITER $$

CREATE TRIGGER ActualizarAsientosDisponibles
AFTER INSERT ON Boletos
FOR EACH ROW
BEGIN
    UPDATE Funciones
    SET Total_Asientos = Total_Asientos - NEW.Cantidad
    WHERE ID_Funcion = NEW.ID_Funcion;
```

```
END$$

DELIMITER ;

#Trigger para Calcular Ingresos Totales por Cine:

DELIMITER $$

CREATE TRIGGER ActualizarIngresosCines
AFTER INSERT ON Boletos
FOR EACH ROW
BEGIN
    DECLARE total_ingresos DECIMAL(10, 2);
    DECLARE id_cine INT;

     SELECT f.ID_Cine INTO id_cine
    FROM Funciones f
    WHERE f.ID_Funcion = NEW.ID_Funcion;

     SELECT COALESCE(SUM(Cantidad * Precio), 0) INTO total_ingresos
    FROM Boletos
    WHERE ID_Funcion = NEW.ID_Funcion;

     INSERT INTO Ingresos_Cines (Id_Cine, Ingreso_Total)
    VALUES (Id_cine, total_ingresos)
    ON DUPLICATE KEY UPDATE Ingreso_Total = total_ingresos;
END$$

DELIMITER ;
```