```
use cursos_medicos;

-- SECCION PARA VISTAS
           ## vista que muestra el nombre completo de los
medicos/doctores y la especialidad que tienen cada uno
                  CREATE VIEW Medicos_Esp as select d.Nombre, d.Apellidos,
esp.Nombre_esp from doctores d
                  join especialidades as esp on esp.Id_Especialidad =
d.Id_Especialidad ;

                  select * from medicos_esp; -- query para mostrar vista

           ## vista para mostrar los alumnos inscritos al curso de
enfermedades metabolicas, donde se pueda ver el contacto y correo del
alumno

                  create view alumnos_cursos
                  as select  nombre_curso, concat(nombre,' ', apellidos)
as Nombre_Completo, telefono, correo from alumno_curso ac
                  inner join cursos as c on ac.id_curso = c.id_curso
                  inner join alumnos as a on a.id_alumno = ac.id_alumno ;

                  select * from alumnos_cursos; -- query para mostrar
vista


           ## en esta vista se obtiene la informacion de cada
curso(temario y descripcion)

                  create view Datos_Curso as select nombre_curso, nombre
as tema, descripcion from cursos c
                  inner join temario as t on t.id_temario = c.id_temario ;

                  select * from Datos_Curso; -- query para mostrar vista

           ## vista que muestra la lista de cursos y el costo de cada 1
                  create view list_cursos as select nombre_curso, costo
from cursos;

                  select * from list_cursos;

           ## vista para mostrar los alumnos que han realizado
comentarios a algun curso y la fecha en que realizaron su comentario
                  create view comentarios_cursos as select concat(
nombre,' ', apellidos) as Nombre_Completo, fecha_comment from alumnos a
                  inner join comentarios as c on c.id_alumno = a.id_alumno
;

            select * from comentarios_cursos; -- query para mostrar vista

-- SECCION PARA FUNCIONES
-- FUNCION PARA REALIZAR LA SUMA DEL COSTO DE 2 CURSOS
DELIMITER ##
```

```
CREATE function suma_precio_cursos(precio1 int, precio2 int)
returns int
deterministic
BEGIN
      declare suma_precios int;
    set suma_precios=precio1+precio2;
    return suma_precios;
END ##
select suma_precio_cursos(2500,3200);
drop function name_doctor;

-- FUNCION PARA MOSTRAR EL NOMBRE DEL MEDICO
DELIMITER ##
CREATE function name_doctor(doctor_user int)
returns varchar (250)
reads sql data
deterministic
BEGIN
      declare datos_name varchar(80);
    select concat(Nombre,',',Apellidos) into datos_name from
cursos_medicos.doctores where Id_Doctor=doctor_user;
    return datos_name;
END ##
select name_doctor(1);



-- SECCION PARA STORE PROCEDURE
-- STORE PROCEDURE PARA ORDENAMIENTO DE TABLA.- el ordenamiento del store
y se habilita opcion para que eñl orden pueda ser de manera descendente o
ascendente.

           DELIMITER $$
           CREATE PROCEDURE Q_ORDENAMIENTO(column1 varchar(200),orden
INT)
           BEGIN
            DECLARE Q_BASE VARCHAR(200);
            DECLARE TIPO_ORDEN VARCHAR(10);
            DECLARE Q_FINAL VARCHAR(200);
            SET Q_BASE = 'SELECT NOMBRE, APELLIDOS, ID_ESPECIALIDAD FROM
cursos_medicos.doctores';

            if orden = 1 THEN
                 SET TIPO_ORDEN = 'asc;';
            else
                 SET TIPO_ORDEN = 'desc;';
            end if;

            if column1 = "" then
                 select 'La columna no puede ser vacia';
            else
                 SELECT concat(Q_BASE,' ORDER BY ',column1,' ',
TIPO_ORDEN) into Q_FINAL;
                 SET @m_orden = Q_FINAL;
```

```
            PREPARE EJECUTAR FROM @m_orden;
            EXECUTE EJECUTAR;
            deallocate prepare EJECUTAR;
      end if;

       END $$
       DELIMITER ;

      CALL Q_ORDENAMIENTO('Apellidos',1);

       -- STORE PROCEDURE PARA INSERTAR Y ELIMINAR ELEMENTO.- el
store cuenta con 2 opciones, las cuales dependiendo de la instruccion
indicada inserta o elimina un registro dentro de la tabla temario

      DELIMITER $$
      CREATE PROCEDURE ACCION_TABLE(
            Accion VARCHAR(200),
            TEM_ID INT,
            NameT varchar(100),
            Desct varchar(100)
             )
      BEGIN
            -- INSERT
            if Accion = "INSERT" THEN
                  INSERT INTO Temario(id_temario,Nombre,
Descripcion)
                  VALUES (TEM_ID,NameT, Desct);
            end if;
            -- DELETE
            if Accion = "DELETE" THEN
                  DELETE FROM Temario
                  WHERE Id_temario = TEM_ID;
            end if;
      END$$

      CALL ACCION_TABLE("INSERT",0,'PROBAR INSERT','PROBAR INSERT');
      -- TABLA PARA VALIDAR MODIFICACIONES
      select *from temario;

-- SECCION PARA TRIGGERS
-- TRIGGERS PARA VALIDAR E INSERTAR REGISTRO DE COMENTARIOS
      -- tabla para log de comentarios

      -- se genera trigger para validar que campo de comentario no
sea vacio en caso de que el campo tenga contenido se genera la insercion
      DELIMITER $$
      CREATE TRIGGER valida_comment
      BEFORE INSERT
      ON comentarios FOR EACH ROW
      BEGIN
            DECLARE MSG_ERROR VARCHAR(70);
            if (NEW.comentario = '') then
```

```
                        SET MSG_ERROR = 'EL COMENTARIO NO PUEDE SER
VACIO';
                        SIGNAL SQLSTATE'45000' SET message_text=
MSG_ERROR;
                  end if;
            END$$
            DELIMITER ;

            -- el trigger insert permite agregar un registro a la tabla
log_comment cada que se ingresa un registro para la tabla comentarios. En
esta tabla se agrega usuario que realiza insercion, fecha y hora
            DELIMITER $$
            CREATE TRIGGER insert_commetarios
            AFTER insert
            ON comentarios FOR EACH ROW
            BEGIN
            insert into log_comment(id_comentario, usuario, fecha, hora)
values(1, session_user(),current_date(), current_time());
            END$$
            DELIMITER ;

            -- query para insertar registro en tabla comnetarios
            insert into comentarios(Comentario, id_curso, Id_alumno,
id_doctor) values('Prueba de comentario', 1, 4, 0);

            select * from comentarios; -- query para ver comentarios
ingresados
            select * from log_comment; -- query para ver los registros en
log_comment realidos en la tabla comentarios

-- TRIGGERS PARA VALIDAR Y ACTUALIZAR CALIFICACION DE ALUMNOS
            -- tabla para log donde se guardaran los datos de quien
modifica la fecha y la hora en que se realiza un cambio

        -- de declara triger que valida que se ingrese una calificacion
permitida para hacer la actualizacion
        DELIMITER $$
            CREATE TRIGGER valida_calificacion
            BEFORE update
            ON Alumno_Curso FOR EACH ROW
            BEGIN
                  DECLARE MSG_ERROR VARCHAR(70);
                  if (NEW.calificacion = 0) then
                        SET MSG_ERROR = 'Ingrese una calificacion valida';
                        SIGNAL SQLSTATE'45000' SET message_text=
MSG_ERROR;
                  end if;
            END$$
            DELIMITER ;
        -- de declara triger para que al actualizar un registro de
calificacion, se genere un log del cambio realizado con los datos
correspondientes a quien realiza el cambio o actualizacion
        DELIMITER $$
            CREATE TRIGGER update_calificacion
```

```
            AFTER update
            ON Alumno_Curso FOR EACH ROW
            BEGIN
                  insert into logs_Calif(calificacion, usuario, fecha,
hora) values(1, session_user(),current_date(), current_time());
            END$$
            DELIMITER ;

         -- query para actualizar calificacion de tabla
         update Alumno_Curso
         set calificacion = 7,
                  id_curso = 1,
             id_alumno = 1
             where id_alumno_curso = 1;

         select *  from Alumno_curso; -- para mostrar tabla
         select *  from logs_calif;     -- para mostrar tabla donde se
guardan los logs del cambio
```