EJERCICIOS PROCEDIMIENTOS ESTRUCTURAS DE CONTROL Y REPETITIVAS.

Realizar en la base de datos PROCEDIMIENTO

1. Escribir un procedimiento que reciba un número entero de entrada y muestre un mensaje indicando si es un número positivo, negativo o cero.

```
DELIMITER //
CREATE PROCEDURE calcular_num_positivo_negativo(IN num int)
BEGIN
    if num < 0 THEN
   SELECT "El numero es negativo" as resultado;
   ELSE
   SELECT "el numero es positivo" as resultado;
   END IF;
end //

DELIMITER ;



call calcular_num_positivo_negativo(5);
call calcular_num_positivo_negativo(-3);
```

2. Escribir un procedimiento que reciba un número real de entrada, que representa el valor de la nota de un alumno, y muestre un mensaje indicando qué nota ha obtenido teniendo en cuenta las siguientes condiciones:

   [0,5) = Insuficiente
   [5,6) = Aprobado
   [6, 7) = Bien
   [7, 9) = Notable
   [9, 10] = Sobresaliente
   En cualquier otro caso la nota no será válida.
   REALIZARLO UTILIZANDO IF

```
DELIMITER //

CREATE PROCEDURE calcular_nota(in nota decimal(4,2))
BEGIN
        if nota >= 0 and nota < 5 THEN
   SELECT "Insuficiente" as resultado;
   ELSEIF nota >= 5 and nota < 6 THEN
   SELECT "Aprobado" as resultado;
```

```
    ELSEIF nota >= 6 and nota < 7 THEN
    SELECT "Bien" as resultado;
    ELSEIF nota >= 7 and nota < 9 THEN
    SELECT "notable" as resultado;
    ELSEIF nota >= 9 and nota <= 10 THEN
    SELECT "sobresaliente" as resultado;
    ELSE SELECT "Nota no valida" as resultado;
    end if;
end //

delimiter ;

call calcular_nota(0);
call calcular_nota(2);
call calcular_nota(3);
call calcular_nota(4);
call calcular_nota(5);
call calcular_nota(6);
call calcular_nota(7);
call calcular_nota(8);
call calcular_nota(9);
call calcular_nota(10);
```

3. Modifica el procedimiento diseñado en el ejercicio anterior para que tenga un parámetro de entrada, con el valor de la nota en formato numérico y un parámetro de salida, con una cadena de texto indicando la nota correspondiente. Realiza el procedimiento haciendo uso de la estructura de control CASE.

```
DELIMITER //


CREATE PROCEDURE calcular_nota2(in nota decimal, out resultado varchar(20))

BEGIN

    CASE
```

```
        WHEN nota >= 0 AND nota < 5 THEN
            SET resultado = 'Insuficiente';
        WHEN nota >= 5 AND nota < 6 THEN
            SET resultado = 'Aprobado';
        WHEN nota >= 6 AND nota < 7 THEN
            SET resultado = 'Bien';
        WHEN nota >= 7 AND nota < 9 THEN
            SET resultado = 'Notable';
        WHEN nota >= 9 AND nota <= 10 THEN
            SET resultado = 'Sobresaliente';
        ELSE
            SET resultado = 'Nota no válida';
    END CASE;

end //

DELIMITER ;

set @nota :=  "";
CALL calcular_nota2(3, @nota)
```

```
SELECT @nota as nota;
```

4. Modificar el procedimiento anterior para que tenga un parámetro de entrada, con el valor un número entero, y un parámetro de salida, con una cadena de caracteres indicando si el número es positivo, negativo o cero.

```
DELIMITER //

CREATE PROCEDURE tipo_numero(IN num INT, OUT resultado VARCHAR(20))
BEGIN
   IF num < 0 THEN
      SET resultado = 'Negativo';
   ELSEIF num > 0 THEN
      SET resultado = 'Positivo';
   ELSE
      SET resultado = 'Cero';
   END IF;
END //

DELIMITER ;

-- Prueba
SET @res := '';
CALL tipo_numero(0, @res);
SELECT @res AS resultado;
```

5. Crear una base de datos llamada ***BD_CUADRADOS*** que contenga una tabla llamada ***CUADRADO***. La tabla cuadrados debe tener dos columnas de tipo INT UNSIGNED, una columna llamada ***numero*** y otra columna llamada ***cuadrado***.

   Crear un procedimiento llamado ***calcular_cuadrados***. Este procedimiento recibe un parámetro de entrada llamado tope de tipo INT UNSIGNED y calculará el valor de los cuadrados de los primeros números naturales hasta el valor introducido como parámetro. El valor de los números y sus cuadrados deberán ser almacenados en la tabla cuadrados, creada previamente.Tener en cuenta que el procedimiento

deberá eliminar el contenido actual de la tabla antes de insertar los nuevos valores de los cuadrados que va a calcular.

```
CREATE DATABASE IF NOT EXISTS BD_CUADRADOS;
USE BD_CUADRADOS;

CREATE TABLE IF NOT EXISTS cuadrados (
   numero INT UNSIGNED,
   cuadrado INT UNSIGNED
);
```

a. Utiliza un bucle WHILE para resolver el procedimiento.

```
DELIMITER //

CREATE PROCEDURE calcular_cuadrados_while(IN tope INT UNSIGNED)
BEGIN
   DECLARE i INT DEFAULT 1;
   DELETE FROM cuadrados;

   WHILE i <= tope DO
      INSERT INTO cuadrados(numero, cuadrado) VALUES (i, i*i);
      SET i = i + 1;
   END WHILE;
END //

DELIMITER ;
```

b. Utiliza un bucle REPEAT para resolver el procedimiento del ejercicio anterior.

```
DELIMITER //

CREATE PROCEDURE calcular_cuadrados_repeat(IN tope INT UNSIGNED)
BEGIN
   DECLARE i INT DEFAULT 1;
   DELETE FROM cuadrados;
```

```
    REPEAT
        INSERT INTO cuadrados(numero, cuadrado) VALUES (i, i*i);
        SET i = i + 1;
    UNTIL i > tope
    END REPEAT;
END //

DELIMITER ;
```

c. Utiliza un bucle LOOP para resolver el procedimiento del ejercicio anterior.

```
DELIMITER //

CREATE PROCEDURE calcular_cuadrados_loop(IN tope INT UNSIGNED)
BEGIN
    DECLARE i INT DEFAULT 1;
    DELETE FROM cuadrados;

    cuadrado_loop: LOOP
        IF i > tope THEN
            LEAVE cuadrado_loop;
        END IF;
        INSERT INTO cuadrados(numero, cuadrado) VALUES (i, i*i);
        SET i = i + 1;
    END LOOP;
END //

DELIMITER ;
```

6. Crea una base de datos llamada ***PARIMPAR*** que contenga una tabla llamada ***PAR*** y otra tabla llamada ***IMPAR.*** Las dos tablas deben tener una única columna llamada ***numero*** y el tipo de dato de esta columna debe ser INT UNSIGNED. Una vez creada la base de datos y las tablas deberá crear un procedimiento llamado

calcular_pares_impares con las siguientes características: El procedimiento recibe un parámetro de entrada llamado tope de tipo INT UNSIGNED y deberá almacenar en la tabla pares aquellos números pares que
existan entre el número 1 y el valor introducido como parámetro. Habrá que realizar la misma operación para almacenar los números impares en la tabla impares.
Tener en cuenta que el procedimiento deberá eliminar el contenido actual de las tablas antes de insertar los nuevos valores.

```
CREATE DATABASE IF NOT EXISTS PARIMPAR;
USE PARIMPAR;

CREATE TABLE IF NOT EXISTS par (
   numero INT UNSIGNED
);

CREATE TABLE IF NOT EXISTS impar (
   numero INT UNSIGNED
);
```

a. Utiliza un bucle WHILE para resolver el procedimiento.

```
DELIMITER //

CREATE PROCEDURE calcular_pares_impares_while(IN tope INT UNSIGNED)
BEGIN
   DECLARE i INT DEFAULT 1;
   DELETE FROM par;
   DELETE FROM impar;

   WHILE i <= tope DO
      IF MOD(i, 2) = 0 THEN
         INSERT INTO par VALUES(i);
      ELSE
         INSERT INTO impar VALUES(i);
      END IF;
      SET i = i + 1;
   END WHILE;
END //

DELIMITER ;
```

b. Utiliza un bucle REPEAT para resolver el procedimiento del ejercicio anterior.

```
DELIMITER //

CREATE PROCEDURE calcular_pares_impares_repeat(IN tope INT
UNSIGNED)
BEGIN
   DECLARE i INT DEFAULT 1;
   DELETE FROM par;
   DELETE FROM impar;

   REPEAT
      IF MOD(i, 2) = 0 THEN
         INSERT INTO par VALUES(i);
      ELSE
         INSERT INTO impar VALUES(i);
      END IF;
      SET i = i + 1;
   UNTIL i > tope
   END REPEAT;
END //

DELIMITER ;
```

c. Utiliza un bucle LOOP para resolver el procedimiento del ejercicio anterior.

```
DELIMITER //

CREATE PROCEDURE calcular_pares_impares_loop(IN tope INT UNSIGNED)
BEGIN
   DECLARE i INT DEFAULT 1;
   DELETE FROM par;
   DELETE FROM impar;

   bucle: LOOP
```

```
        IF i > tope THEN
           LEAVE bucle;
        END IF;

        IF MOD(i, 2) = 0 THEN
           INSERT INTO par VALUES(i);
        ELSE
           INSERT INTO impar VALUES(i);
        END IF;

        SET i = i + 1;
     END LOOP;
END //

DELIMITER ;
```