Bases de Datos

Pavel Miron

UD4 - Práctica 5.

Índice

Contenido

1.Primero de primaria escribe un procedimiento que devuelva como resultado la suma de dos números enteros los cuales se le pasan como parámetros

```
-- 1
DELIMITER $$
create procedure suma (
    in num1 int,
    in num2 int,
    out resultado int
)
begin
    set resultado = num1 + num2;
END $$
--
call suma (2, 2, @resultado);
select @resultado;
```

Result Grid

	@resultado
▶	4

2. ¿Segundo de primaria? escribe un procedimiento que devuelva como resultados la suma y la multiplicación de dos números enteros. El resultado de la multiplicación debe devolverse en la misma variable que determina el segundo número

```
-- 2
DELIMITER $$
create procedure procedimiento (
    in num1 int,
    inout num2 int
)
begin

    select num1 + num2 AS suma;
    set num2 = num1 * num2;

END $$
--
set @num2 = 5;
call procedimiento (3, @num2);
select @num2 AS multiplicacion;
```

Result Grid

	multiplicacion
▶	15

3. Recuperación del primer ciclo rehaz la actividad 1 con una función en lugar de un procedimiento

```
-- 3
DELIMITER $$
create function funcion (
    num1 int,
    num2 int
) returns int
deterministic
begin
    return resultado = num1 + num2;
END $$
--
select funcion(7, 8) as resultado;
```

4. Día sin IVA crea un procedimiento que reciba como parámetro un precio y calcule su precio sin IVA (considera IVA al 21%), devolviéndolo en una variable

```
DELIMITER $$
create procedure precio_sin_iva(
    in precio decimal(10,2),
    out precio_sin_iva decimal(10,2)
)
begin
    set precio_sin_iva = precio / 1.21;
END $$

--
call precio_sin_iva(121, @precio_sin_iva);
select @precio_sin_iva AS precio_sin_iva;
```

Result Grid

precio_sin_iva
100.00

5. Del calendario no Crear una función para mostrar el día de la semana según el valor de entrada numérico: 1 para lunes, 2 para martes, etc…

```
-- 5
DELIMITER $$
create function dia_semana (
    num_dia int
) returns varchar(20)
deterministic
begin
    declare dia varchar(20);

    case num_dia
        when 1 then set dia = 'lunes';
        when 2 then set dia = 'martes';
        when 3 then set dia = 'miercoles';
        when 4 then set dia = 'jueves';
        when 5 then set dia = 'viernes';
        when 6 then set dia = 'sabado';
        when 7 then set dia = 'domingo';
        else set dia = 'invalido';
    end case;

    return dia;
END$$

--
select dia_semana(2) as dia;
```

Result Grid

	dia
▶	martes

6. Esto compila… y calcula crear una función calculadora que realice operaciones con dos números decimales. La operación a realizar depende de un tercer parámetro que puede ser suma, resta, mult o div

```
-- 6
DELIMITER $$
create function calculadora(
    num1 decimal(10,2),
    num2 decimal(10,2),
    operacion varchar(10)
) returns decimal(10,2)
deterministic
begin
    declare resultado decimal(10,2);

    case operacion
        when 'suma' then set resultado = num1 + num2;
        when 'resta' then set resultado = num1 - num2;
        when 'mult' then set resultado = num1 * num2;
        when 'div' then
            if num2 = 0 then
                signal sqlstate '45000' set message_text = 'división por cero no permitida';
            else
                set resultado = num1 / num2;
            end if;
        else
            signal sqlstate '45000' set message_text = 'operación no válida';
    end case;

    return resultado;
END $$

--
select calculadora(10, 2, 'div') as resultado;
```

Result Grid

	resultado
▶	5.00

7. El factorial crea un procedimiento que calcule el factorial de N. N será un número proporcionado por el usuario como argumento al procedimiento

```
-- 7
DELIMITER $$
create procedure calcular_factorial(
    in n int,
    out resultado int
)
begin
    declare i int;
    set resultado = 1;
    set i = 1;

    while i <= n do
        set resultado = resultado * i;
        set i = i + 1;
    end while;
END $$

--
call calcular_factorial(5, @factorial);
select @factorial as '5!';
```

Result Grid

	5!
▶	120

8. El mismo saco crea un procedimiento que introduce en una tabla denominada “impares” los primeros 50 números impares

```
-- 8
DELIMITER $$
create procedure impares()
begin
    declare contador int default 1;
    declare num_impar int default 1;

    create table if not exists impares (
        id int primary key,
        num_impar int
    );

    truncate table impares;

    while contador <= 50 do
        insert into impares values (contador, num_impar);
        set contador = contador + 1;
        set num_impar = num_impar + 2;
    end while;
END $$

--
call impares();
select count(*) as total_impares from impares;
```

Result Grid

	total_impares
▶	50