

TALLER BASE DE DATOS

PRESENTADO POR:

DARWIN STEVEN GOMEZ

DIRIGIDO A:

CESAR CUELLAR

CENTRO DE TELEINFORMATICA Y PRODUCCION INDUSTRIAL

DESARROLLO DE SOFTWARE

SENA REGIONAL CAUCA

POPAYAN CAUCA

2024

Contenido

Ejercicios a Desarrollar para entregar como evidencia	3
1. Crear un procedimiento de nombre día_de_la_semana que reciba como parámetro de entrada un valor numérico que represente un día de la semana y que devuelva una cadena de caracteres con el nombre del día de la semana correspondiente. Por ejemplo, para el valor de entrada 1 debería devolver la cadena lunes. Resuelva el procedimiento haciendo uso de la estructura de control IF.	3
2. Crear un procedimiento de nombre calcular_valores_pago, el cual recibe como parámetro de entrada una forma de pago, que será una cadena de caracteres (Ejemplo: PayPal, Transferencia, etc). Y devuelva como salida los siguientes valores teniendo en cuenta la forma de pago seleccionada como parámetro de entrada:	4
• el pago de máximo valor,	4
• el pago de mínimo valor,	4
• el valor medio de los pagos realizados,	4
• la suma de todos los pagos,	4
• el número de pagos realizados para esa forma de pago.	4
• Deberá hacer uso de la tabla pago de la base de datos jardineria.	4

Ejercicios a Desarrollar para entregar como evidencia

1. Crear un procedimiento de nombre `día_de_la_semana` que reciba como parámetro de entrada un valor numérico que represente un día de la semana y que devuelva una cadena de caracteres con el nombre del día de la semana correspondiente. Por ejemplo, para el valor de entrada 1 debería devolver la cadena `lunes`. Resuelva el procedimiento haciendo uso de la estructura de control IF.

```
DELIMITER //
```

```
create procedure día_de_la_semana(in día int, out nombre_día varchar(10))
```

```
begin
```

```
    if día = 1 then
```

```
        set nombre_día = 'lunes';
```

```
    elseif día = 2 then
```

```
        set nombre_día = 'martes';
```

```
    elseif día = 3 then
```

```
        set nombre_día = 'miércoles';
```

```
    elseif día = 4 then
```

```
        set nombre_día = 'jueves';
```

```
    elseif día = 5 then
```

```
        set nombre_día = 'viernes';
```

```
    elseif día = 6 then
```

```
        set nombre_día = 'sábado';
```

```
    elseif día = 7 then
```

```
        set nombre_día = 'domingo';
```

```
    else
```

```
        set nombre_día = 'día inválido';
```

```
    end if;
```

```
end //
```

```
DELIMITER ;
```

Action Output				
#	Time	Action	Message	Duration / Fetch
1	21:06:59	create procedure día_de_la_semana(in día int, out nombre_día varchar(10)) begin if día = 1 then...	Error Code: 1304. PROCEDURE día_de_la_semana already exists	0.000 sec
2	21:07:11	create procedure día_de_la_semana(in día int, out nombre_día varchar(10)) begin if día = 1 then...	Error Code: 1304. PROCEDURE día_de_la_semana already exists	0.000 sec

```

1 • set @dia = 1;
2 • call dia_de_la_semana(@dia, @nombre_dia);
3 • select @nombre_dia;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

@nombre_dia
lunes

Result 2 x

Output

Action Output

#	Time	Action	Message
1	21:06:59	create procedure dia_de_la_semana(in dia int, out nombre_dia varchar(10)) begin if dia = 1 the...	Error Code: 1304. PROCEDURE dia_de_la_semana already exists
2	21:07:11	create procedure dia_de_la_semana(in dia int, out nombre_dia varchar(10)) begin if dia = 1 the...	Error Code: 1304. PROCEDURE dia_de_la_semana already exists
3	21:08:34	set @dia = 1	0 row(s) affected
4	21:08:34	call dia_de_la_semana(@dia, @nombre_dia)	0 row(s) affected
5	21:08:34	select @nombre_dia LIMIT 0, 1000	1 row(s) returned

2. Crear un procedimiento de nombre `calcular_valores_pago`, el cual recibe como parámetro de entrada una forma de pago, que será una cadena de caracteres (Ejemplo: PayPal, Transferencia, etc.). Y devuelva como salida los siguientes valores teniendo en cuenta la forma de pago seleccionada como parámetro de entrada:

- el pago de máximo valor,
- el pago de mínimo valor,
- el valor medio de los pagos realizados,
- la suma de todos los pagos,
- el número de pagos realizados para esa forma de pago.
- Deberá hacer uso de la tabla `pago` de la base de datos `jardineria`.

```

delimiter //
> create procedure calcular_valores_pago(
    in forma_pago varchar(40),
    out max_valor decimal(15, 2),
    out min_valor decimal(15, 2),
    out valor_medio decimal(15, 2),
    out suma_valores decimal(15, 2),
    out numero_pagos int
)
> begin
    select max(total) into max_valor
    from pago
    where forma_pago = forma_pago;
    select min(total) into min_valor
    from pago
    where forma_pago = forma_pago;
    select avg(total) into valor_medio
    from pago
    where forma_pago = forma_pago;
    select sum(total) into suma_valores
    from pago
    where forma_pago = forma_pago;
    select count(*) into numero_pagos
    from pago
    where forma_pago = forma_pago;
end //
delimiter ;

```



1 21:21:44 CREATE PROCEDURE calcular_valores_pago(IN forma_pago VARCHAR(40), OUT ... Error Code: 1304: PROCEDURE calcular_valores_pago already exists

0.000 sec

```

1 • set @forma_pago = 'paypal';
2 set @max_valor = 0;
3 set @min_valor = 0;
4 set @valor_medio = 0;
5 set @suma_valores = 0;
6 set @numero_pagos = 0;
7
8 call calcular_valores_pago(@forma_pago, @max_valor, @min_valor, @valor_medio, @suma_valores, @numero_pagos);
9

```

Result Grid

	max_valor	min_valor	valor_medio	suma_valores	numero_pagos
▶	23794.00	232.00	7613.08	197940.00	26

Result 2 ×

Output

Action Output

#	Time	Action	Message
✓ 7	21:21:54	SET @numero_pagos = 0	0 row(s) affected
✓ 8	21:21:54	CALL calcular_valores_pago(@forma_pago, @max_valor, @min_valor, @valor_medio, ...	1 row(s) affected
✓ 9	21:21:54	SELECT @max_valor AS max_valor, @min_valor AS min_valor, @valor_medio AS valor...	1 row(s) returned
✓ 10	21:25:34	set @forma_pago = 'paypal'	0 row(s) affected
✓ 11	21:25:34	set @max_valor = 0	0 row(s) affected
✓ 12	21:25:34	set @min_valor = 0	0 row(s) affected

3. Crear una base de datos llamada procedimientos que contenga una tabla llamada pares y otra tabla llamada impares. Las dos tablas deben tener única columna llamada número 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 el valor introducido como parámetro. Habrá que realizar la misma operación para almacenar los números impares en la tabla impares. Tenga en cuenta que el procedimiento deberá eliminar el contenido actual de las tablas antes de insertar los nuevos valores. Utilice un bucle WHILE para resolver el procedimiento.

```
1 • create database procedimientos;
2 • use procedimientos;
3 • create table pares (
4     numero int unsigned
5 );
6 • create table impares (
7     numero int unsigned
8 );
9
```


Output			
Action Output			
#	Time	Action	
✓ 1	21:29:47	create database procedimientos	
✓ 2	21:29:47	use procedimientos	
✓ 3	21:29:47	create table pares (numero int unsigned)	
✓ 4	21:29:47	create table impares (numero int unsigned)	

```

1 delimiter //
2 • create procedure calcular_pares_impares(in tope int unsigned)
3 begin
4     declare i int unsigned default 1;
5     delete from pares;
6     delete from impares;
7     while i <= tope do
8         if mod(i, 2) = 0 then
9             insert into pares (numero) values (i);
10        else
11            insert into impares (numero) values (i);
12        end if;
13        set i = i + 1;
14    end while;
15 end //
16 delimiter ;

```

Output

 Action Output

#	Time	Action	Message
✓ 1	21:31:46	create procedure calcular_pares_impares(in tope int unsigned) begin declare i int unsigne...	0 row(s) affected


```

1      #modo seguro desactivado
2      •   set SQL_SAFE_UPDATES = 0;
3
4      •   call calcular_pares_impares(10);
5
6      •   select * from procedimientos.pares;
7
8      •   select * from procedimientos.impares;
9

```

Result Grid		Filter Rows:	Export:
	numero		
▶	2		
	4		
	6		
	8		
	10		

```

8      •   select * from procedimientos.impares;
9

```

Result Grid		Filter Rows:	Export:
	numero		
▶	1		
	3		
	5		
	7		
	9		

4. Crear una función de nombre obtener_mes que reciba como parámetro de entrada un valor numérico que represente un mes del año y que devuelva una cadena de caracteres con el nombre del mes correspondiente. Por ejemplo, para el valor de entrada 1 debería devolver la cadena Enero, si el valor de entrada es 12 debería devolver Diciembre. Utilizar la estructura de control CASE

```
DELIMITER //
CREATE FUNCTION obtener_mes(mes INT) RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
    RETURN CASE mes
        WHEN 1 THEN 'Enero'
        WHEN 2 THEN 'Febrero'
        WHEN 3 THEN 'Marzo'
        WHEN 4 THEN 'Abril'
        WHEN 5 THEN 'Mayo'
        WHEN 6 THEN 'Junio'
        WHEN 7 THEN 'Julio'
        WHEN 8 THEN 'Agosto'
        WHEN 9 THEN 'Septiembre'
        WHEN 10 THEN 'Octubre'
        WHEN 11 THEN 'Noviembre'
        WHEN 12 THEN 'Diciembre'
        ELSE 'Mes inválido'
    END;
END //
DELIMITER ;
```

1 21:47:22 CREATE FUNCTION obtener_mes(mes INT) RETURNS VARCHAR(20) DETERMINISTIC ... 0 row(s) affected

0.016 sec

```

1 • SELECT obtener_mes(1) AS nombre_mes;
2 SELECT obtener_mes(12) AS nombre_mes;
3 SELECT obtener_mes(13) AS nombre_mes;

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	nombre_mes			
►	Mes inválido			

Result 13	Result 14	Result 15	×
Output			
Action Output			
#	Time	Action	Message
✓ 14	21:50:38	SELECT obtener_mes(13) AS nombre_mes LIMIT 0, 1000	1 row(s) returned
✓ 15	21:50:41	SELECT obtener_mes(1) AS nombre_mes LIMIT 0, 1000	1 row(s) returned
✓ 16	21:50:41	SELECT obtener_mes(12) AS nombre_mes LIMIT 0, 1000	1 row(s) returned
✓ 17	21:50:41	SELECT obtener_mes(13) AS nombre_mes LIMIT 0, 1000	1 row(s) returned

5. Crear una función de nombre `obtener_total_pagos_mes_año`, la cula recibe como parámetros de entrada el mes y el año y como resultado devuelva la suma del total de pagos realizados en ese mes y año. Los valores de entrada deben ser de tipo entero, y el tipo de dato que retorna la función debe ser del mismo tipo de datos del campo total de la tabla pago de la base de datos jardinería.

```

DELIMITER //
CREATE FUNCTION obtener_total_pagos_mes_año(mes INT, año INT)
RETURNS DECIMAL(15, 2)
DETERMINISTIC
BEGIN
    DECLARE total_pagos DECIMAL(15, 2);
    SELECT SUM(total) INTO total_pagos
    FROM pago
    WHERE MONTH(fecha_pago) = mes AND YEAR(fecha_pago) = año;
    IF total_pagos IS NULL THEN
        SET total_pagos = 0;
    END IF;
    RETURN total_pagos;
END //
DELIMITER ;

```

tion Output

Time	Action	Message
1 22:42:21	CREATE FUNCTION obtener_total_pagos_mes_año(mes INT, año INT) RETURNS DEC...	Error Code: 1304. FUNCTION obtener_total_pagos_mes_año already exists

```

SELECT obtener_total_pagos_mes_año(1, 2009) AS total_pagos;
SELECT obtener_total_pagos_mes_año(12, 2008) AS total_pagos;

```

6. Crear una función de nombre `cantidad_total_de_productos_vendidos`, la cual recibe como parámetro de entrada el código de un producto y como resultado devuelva la cantidad total de productos que se han vendido con ese código. Utilizar base datos `jardinería`

```
5 begin
6     declare total_vendido int;
7     select coalesce(sum(cantidad), 0) into total_vendido
8     from detalle_pedido
9     where codigo_producto = codigo_producto_input;
10    return total_vendido;
11 end //
12 delimiter ;
13
14 • SELECT cantidad_total_de_productos_vendidos('FR-67') AS total_vendido;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	total_vendido
▶	285

Result 2 x

Output

Action Output

#	Time	Action	Message
✓ 1	22:53:11	create function cantidad_total_de_productos_vendidos(codigo_producto_input varchar(15...	0 row(s) affected
✓ 2	22:53:29	SELECT cantidad_total_de_productos_vendidos('S10_1949') AS total_vendido LIMIT 0, 1...	1 row(s) returned
✓ 3	22:56:06	SELECT cantidad_total_de_productos_vendidos('FR-67') AS total_vendido LIMIT 0, 1000	1 row(s) returned