



**PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL**  
**FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES**  
**ACTIVIDAD PROGRAMACIÓN BASES DE DATOS**  
**PROCEDIMIENTOS ALMACENADOS – FUNCIONES Y TRIGGERS**

Presentado a:  
Instructor, CESAR MARINO CUELLAR

Por Aprendiz:  
Lilian Socorro Anacona

Ficha: 2874057

Competencia: Bases de Datos

Tecnólogo en Análisis y Desarrollo de Software  
Servicio Nacional de Aprendizaje SENA  
Centro de Teleinformática y Producción Industrial Regional Cauca  
Popayán 4 de noviembre del año 2024



## Actividad: Programación en SQL (Funciones, Procedimientos Almacenados, Triggers)

### 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.

```
1 DELIMITER //
```

```
2
```

```
3 • create procedure día_de_la_semana (in dia int, out nombre_dia varchar(30))
```

```
4 begin
```

```
5 IF dia=1 then set nombre_dia="lunes";
```

```
6 elseif dia=2 then set nombre_dia="martes";
```

```
7 elseif dia=3 then set nombre_dia="miercoles";
```

```
8 elseif dia=4 then set nombre_dia="jueves";
```

```
9 elseif dia=5 then set nombre_dia="viernes";
```

```
10 elseif dia=6 then set nombre_dia="sabado";
```

```
11 elseif dia=7 then set nombre_dia="domingo";
```

```
12 else set nombre_dia="dato invalido";
```

```
13 end if;
```

```
14 end //
```

```
15
```

```
16 DELIMITER ;
```

```
17 • call día_de_la_semana (3, @resultado);
```

```
18 • select @resultado;
```



```
16 DELIMITER ;
17 • call día_de_la_semana (3, @resultado);
18 • select @resultado;
19
20
21
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
@resultado			
► miercoles			

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**.



```
22 DELIMITER //
```

```
23 • CREATE PROCEDURE calcular_valores_pago (IN forma_pago VARCHAR(60))
```

```
24 BEGIN
```

```
25     SELECT MAX(total) AS max_pago,
```

```
26     MIN(total) as min_pago,
```

```
27     avg(total) as promedio_pago,
```

```
28     sum(total) as suma_pagos,
```

```
29     count(*) as num_pagos
```

```
30     from pago p where p.forma_pago= forma_pago;
```

```
31 END //
```

```
32
```

```
33 DELIMITER ;
```

```
34 • call calcular_valores_pago ("PayPal");
```

```
35
```

```
33 DELIMITER ;
```

```
34 • call calcular_valores_pago ("PayPal");
```

```
35
```

Result Grid   Filter Rows:   Export:   Wrap Cell Content:					
	max_pago	min_pago	promedio_pago	suma_pagos	num_pagos
▶	20000.00	232.00	7156.500000	171756.00	24

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.



```
38 • create database procedimientos;
39 • use procedimientos;
40
41 • create table pares(
42     numero int unsigned
43 );
44
45 • create table impares(
46     numero int unsigned
47 );
48
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
18	21:04:41	CREATE PROCEDURE calcular_valores_pago (l...	Error Code: 1304. PROCEDURE calcular_valores...	0.016 sec
19	21:04:50	call calcular_valores_pago ("PayPal")	1 row(s) returned	0.000 sec / 0.000 sec
20	21:05:57	create database procedimientos	1 row(s) affected	0.016 sec
21	21:06:09	use procedimientos	0 row(s) affected	0.000 sec
22	21:07:33	create table pares(numero int unsigned )	0 row(s) affected	0.046 sec
23	21:07:33	create table impares(numero int unsigned )	0 row(s) affected	0.016 sec

```
49 DELIMITER //
50
51 • create procedure calcular_pares_impares (in tope int unsigned)
52 begin
53     declare i int default 1;
54     delete from pares;
55     delete from impares;
56     while i <= tope do
57         if i %2=0 then
58             insert into pares (numero) values (i);
59         else
60             insert into impares (numero) values (i);
61         end if;
62         set i= i+1;
63     end while;
64 end //
65
66 DELIMITER ;
```



```
68 • call calcular_pares_impares (10);
69 • use procedimientos;
70
71 • select * from pares;
```

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

	numero
▶	2
	4
	6
	8
	10

```
71 • select * from pares;
72 • select * from impares;
```

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

	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 debera devolver Diciembre. Utilizar la estructura de control **CASE**.



```
75 DELIMITER //
76 • create function obtener_mes (mes int) returns varchar (25) deterministic
77 begin
78 return case
79 when mes = 1 then "enero"
80 when mes = 2 then "febrero"
81 when mes = 3 then "marzo"
82 when mes = 4 then "abril"
83 when mes = 5 then "mayo"
84 when mes = 6 then "junio"
85 when mes = 7 then "julio"
86 when mes = 8 then "agosto"
87 when mes = 9 then "septiembre"
88 when mes = 10 then "octubre"
89 when mes = 11 then "noviembre"
90 when mes = 12 then "diciembre"
91 else "el dato no es valido"
92 end;
93 end //
94 DELIMITER ;
```

```
95 • select obtener_mes (5);
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
obtener_mes (5)				
▶ mayo				

5. Crear una función de nombre **obtener\_total\_pagos\_mes\_año**, la cual 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.



```
103 DELIMITER //
104 • create function obtener_total_pagos_mes_año (mes int, anio int) returns decimal (10,2) determi
105 begin
106     declare total_pagos decimal (10,2);
107     select sum(total) into total_pagos
108     from pago where month(fecha_pago)= mes and year(fecha_pago)= anio;
109     return total_pagos;
110 end //
111 DELIMITER ;
```

```
113 • select obtener_total_pagos_mes_año (10, 2007 );
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	obtener_total_pagos_mes_año (10, 2007)			
▶	3321.00			

6. Crear una función de nombre **int** , 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.

```
119 DELIMITER //
120 • drop function if exists cantidad_total_de_productos_vendidos //
121 • create function cantidad_total_de_productos_vendidos (codigo_producto varchar(15)) returns int deterministic
122 begin
123     declare total int;
124     select sum(cantidad) into total from detalle_pedido dp
125     where dp.codigo_producto = codigo_producto;
126     return total;
127 end //
128 DELIMITER ;
```

```
129
130 • select cantidad_total_de_productos_vendidos ("OR-241");
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	cantidad_total_de_productos_vendidos ("OR-241")			
▶	25			

7. Crear una tabla que se llame **notificaciones** en la base de datos **jardinería** que tenga las siguientes columnas:





- **id** (entero sin signo, autoincremento y clave primaria)
- **fecha\_hora**: registro fecha y hora del pago (datetime)
- **total**: el valor del pago (real)
- **codigo\_cliente**: código del cliente que realiza el pago (entero)

Escriba un **trigger** que nos permita llevar un control de los pagos que van realizando los clientes.

Los detalles de implementación del trigger son los siguientes:

- Nombre del trigger: **trigger\_notificar\_pago**
- Se ejecuta sobre la tabla pago.
- Se ejecuta *después* de hacer la inserción de un pago.
- Cada vez que un cliente realice un pago (es decir, se hace una inserción en la tabla pago), el *trigger* deberá insertar un nuevo registro en una tabla llamada notificaciones.

```
133 • use jardineria;
134
135 • create table notificaciones (
136     id int unsigned auto_increment primary key,
137     fecha_hora datetime not null,
138     total numeric(10,2) not null,
139     codigo_cliente int not null
140 );
141
142 DELIMITER //
143 • create trigger trigger_notificar_pago
144 after insert on pago
145 for each row
146 begin
147     insert into notificaciones (fecha_hora, total, codigo_cliente)
148     values (new.fecha_pago, new.total, new.codigo_cliente);
149 end //
150 DELIMITER ;
```

Output

Action Output				
#	Time	Action	Message	Duration / Fetch
✓ 19	22:09:07	select cantidad_total_de_productos_vendidos ("...	1 row(s) returned	0.000 sec / 0.000 sec
✓ 20	22:10:03	select cantidad_total_de_productos_vendidos ("...	1 row(s) returned	0.000 sec / 0.000 sec
✓ 21	22:17:33	use jardineria	0 row(s) affected	0.000 sec
✓ 22	22:22:15	create table notificaciones ( id int unsigned auto_j...	0 row(s) affected	0.031 sec
✗ 23	22:22:36	create trigger trigger_notificar_pago after insert on...	Error Code: 1054. Unknown column 'fecha' in 'NE...	0.000 sec
✓ 24	22:23:21	create trigger trigger_notificar_pago after insert on...	0 row(s) affected	0.016 sec



```
151 • select * from cliente;
152 • insert into pago (fecha_pago, total, codigo_cliente, forma_pago, id_transaccion) values (now(),3500.00, 10, "PayPal","1234");
153
154 • select * from notificaciones;
```

Result Grid				
Filter Rows:				
Edit: Export/Import: Wrap Cell Content:				
	id	fecha_hora	total	codigo_cliente
▶	1	2024-12-04 00:00:00	3500.00	10
*		NULL	NULL	NULL

8. Crear una **base de datos** llamada test que contenga una **tabla** llamada alumnos con las siguientes columnas.

#### Tabla alumnos:

- id (entero sin signo)
- nombre (cadena de caracteres)
- apellido1 (cadena de caracteres)
- apellido2 (cadena de caracteres)
- email (cadena de caracteres)

```
158 • drop database test;
159 • create database test;
160 • use test;
161
162 • create table alumnos(
163     id int unsigned primary key auto_increment,
164     nombre varchar(50) not null,
165     apellido1 varchar(50) not null,
166     apellido2 varchar(50) not null,
167     email varchar (255)
168 );
```

Crear un procedimiento llamado **crear\_email** que dados los parámetros de entrada: nombre, apellido1, apellido2 y dominio, cree una dirección de email y la devuelva como salida.

- Procedimiento: **crear\_email**
- Parámetros de Entrada:
  - nombre (cadena de caracteres)
  - apellido1 (cadena de caracteres)
  - apellido2 (cadena de caracteres)
  - dominio (cadena de caracteres)
- Parámetros de Salida:



- email (cadena de caracteres)

devuelva una dirección de correo electrónico con el siguiente formato:

- El primer carácter del parámetro nombre.
- Los tres primeros caracteres del parámetro apellido1.
- Los tres primeros caracteres del parámetro apellido2.
- El carácter @.
- El dominio pasado como parámetro.
- La dirección de email debe estar en minúsculas.

```
171 DELIMITER //
172 • create procedure crear_email (in nombre varchar(50), in apellido1 varchar(50), in apellido2 varchar(50), in dominio varchar(50), out email varchar(255))
173 begin
174 set email = lower(concat(left(nombre,1),left(apellido1,3), left(apellido2,3),"@",dominio));
175 end//
176 DELIMITER ;
177
```

Una vez creada la tabla escriba **un trigger** con las siguientes características:

- Nombre del Trigger: **trigger\_crear\_email\_before\_insert**
  - Se ejecuta sobre la tabla alumnos.
  - Se ejecuta *antes* de una operación de *inserción*.
  - Si el nuevo valor del email que se quiere insertar es NULL, entonces se le creará automáticamente una dirección de email y se insertará en la tabla.
  - Si el nuevo valor del email no es NULL se guardará en la tabla el valor del email.

```
180 DELIMITER //
181 • create trigger trigger_crear_email_before_insert
182 before insert on alumnos
183 for each row
184 begin
185 if new.email is null then
186 call crear_email(new.nombre, new.apellido1, new.apellido2, "default.com",@new_email);
187 set new.email=@new_email;
188 end if;
189 end //
190 DELIMITER ;
```

Output

Action Output

#	Time	Action	Message
✓ 33	22:38:20	create database test	1 row(s) affected
✓ 34	22:38:31	use test	0 row(s) affected
✓ 35	22:42:39	create table alumnos( id int unsigned not null, nombre varchar(50) not null, apellido1 varchar(50) not null, apellid...	0 row(s) affected
✓ 36	22:53:25	create procedure crear_email (in nombre varchar(50), in apellido1 varchar(50), in apellido2 varchar(50), in domi...	0 row(s) affected
✓ 37	22:56:24	alter table alumnos modify column email varchar(255) null	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 38	23:04:42	create trigger trigger_crear_email_before_insert before insert on alumnos for each row begin if new.email is null...	0 row(s) affected



```
191 • insert into alumnos(nombre, apellido1, apellido2,email) values("lilian", "anacona", "narvaez", null);
192 • select * from alumnos;
```

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	id	nombre	apellido1	apellido2	email
▶	1	lilian	anacona	narvaez	lananar@default.com
•	NULL	NULL	NULL	NULL	NULL