



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL
FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES
ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

Programación en SQL (Funciones, Procedimientos Almacenados, Triggers)

Presentado a: Cesar Marino Cuellar ingeniero

Por Aprendiz: Arnoby Santiago Quilindo

Ficha: 2874057

Competencia: Construcción Del software
bases de datos

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL
FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES
ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

Contenido

Actividad: Programación en SQL (Funciones, Procedimientos Almacenados, Triggers).....	3
Material de Apoyo:.....	3
Ejercicios a Desarrollar para entregar como evidencia.....	4
1. Crear un procedimiento de nombre día_de_la_semana	4
2. Crear un procedimiento de nombre calcular_valores_pago	6
3. Crear una base de datos llamada procedimientos	11
4. Crear una función de nombre obtener_mes	13
5. Crear una función de nombre obtener_total_pagos_mes_año	16
6. Crear una función de nombre cantidad_total_de_productos_vendidos	18
7. Crear una tabla que se llame notificaciones en la base de datos jardinería	20
8. Crear una base de datos llamada test	22
Tabla alumnos:	22
CONTROL DEL DOCUMENTO	24

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

GF



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL
FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES
ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

PROCEDIMIENTOS ALMACENADOS – FUNCIONES Y TRIGGERS

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

1. Revisar material de apoyo
2. Hacer ejercicios propuestos en el ambiente de formación
3. Realizar ejercicios para entregar como evidencia.

Material de Apoyo:

- Unidad 12. Triggers, procedimientos y funciones en MySQL.
<https://josejuansanchez.org/bd/unidad-12-teoria/index.html>
- Material publicado en Microsoft Teams.

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

GF



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES

ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

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.

The screenshot shows the SQL Developer interface with a script titled 'día de la semana'. The script contains the following SQL code:

```
1 use procedimientos;
2 show databases;
3
4 drop procedure if exists día_de_la_semana;
5 DELIMITER $$
6
7 create PROCEDURE día_de_la_semana(in día int, out nombre_día varchar(10))
8 begin
9     if día = 1 then
10         set nombre_día = 'lunes';
11     elseif día = 2 then
12         set nombre_día = 'martes';
13     elseif día = 3 then
14         SET nombre_día = 'miércoles';
15     elseif día = 4 then
16         set nombre_día = 'jueves';
17     elseif día = 5 then
18         set nombre_día = 'viernes';
19     elseif día = 6 then
20         set nombre_día = 'sábado';
21     elseif día = 7 then
22         set nombre_día = 'domingo';
23     else
24         set nombre_día = 'Día invalido';
25     end if;
26 end if;
```

The left sidebar shows the 'SCHEMAS' pane with 'procedimientos' selected. The bottom pane shows the 'Output' window with the following log:

#	Time	Action	Message	Duration / Fetch
23	10:50:55	show databases	24 row(s) returned	0.000 sec / 0.000 sec
24	10:51:04	use procedimientos	0 row(s) affected	0.000 sec



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES

ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

The screenshot shows the SQL Developer interface with the 'dia_de_la_semana' stored procedure being created. The procedure takes an integer input for the day of the week and returns the corresponding day name as a varchar(10). The code is as follows:

```
drop procedure if exists dia_de_la_semana;
create PROCEDURE dia_de_la_semana(in dia int, out nombre_dia varchar(10))
begin
  if dia = 1 then
    set nombre_dia = 'lunes';
  elseif dia = 2 then
    set nombre_dia = 'martes';
  elseif dia = 3 then
    set nombre_dia = 'miércoles';
  elseif dia = 4 then
    set nombre_dia = 'jueves';
  elseif dia = 5 then
    set nombre_dia = 'viernes';
  elseif dia = 6 then
    set nombre_dia = 'sábado';
  elseif dia = 7 then
    set nombre_dia = 'domingo';
  else
    set nombre_dia = 'Día invalido';
  end if;
end $$
DELIMITER ;
```

The output window shows the execution results:

#	Time	Action	Message	Duration / Fetch
25	10:51:54	drop procedure if exists dia_de_la_semana	0 row(s) affected	0.016 sec
26	10:51:54	create PROCEDURE dia_de_la_semana(in dia int, out nombre_dia varchar(10)) begin if dia = 1 then set n...	0 row(s) affected	0.000 sec

The screenshot shows the SQL Developer interface with the 'dia_de_la_semana' stored procedure being executed. The code is as follows:

```
DELIMITER ;
SHOW PROCEDURE STATUS WHERE Name = 'dia_de_la_semana';
CALL dia_de_la_semana(3, @nombre_dia);
SELECT @nombre_dia;
```

The output window shows the execution results:

#	Time	Action	Message	Duration / Fetch
22	10:18:55	use procedimientos	0 row(s) affected	0.000 sec
23	10:50:55	show databases	24 row(s) returned	0.000 sec / 0.000 sec
24	10:51:04	use procedimientos	0 row(s) affected	0.000 sec
25	10:51:54	drop procedure if exists dia_de_la_semana	0 row(s) affected	0.016 sec
26	10:51:54	create PROCEDURE dia_de_la_semana(in dia int, out nombre_dia varchar(10)) begin if dia = 1 then set n...	0 row(s) affected	0.000 sec
27	10:52:48	SHOW PROCEDURE STATUS WHERE Name = 'dia_de_la_semana'	1 row(s) returned	0.015 sec / 0.000 sec
28	10:52:48	CALL dia_de_la_semana(3, @nombre_dia)	0 row(s) affected	0.000 sec
29	10:52:48	SELECT @nombre_dia LIMIT 0, 10000	1 row(s) returned	0.000 sec / 0.000 sec

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

GF



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES

ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

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**.

The screenshot displays the SQL Developer environment with the following components:

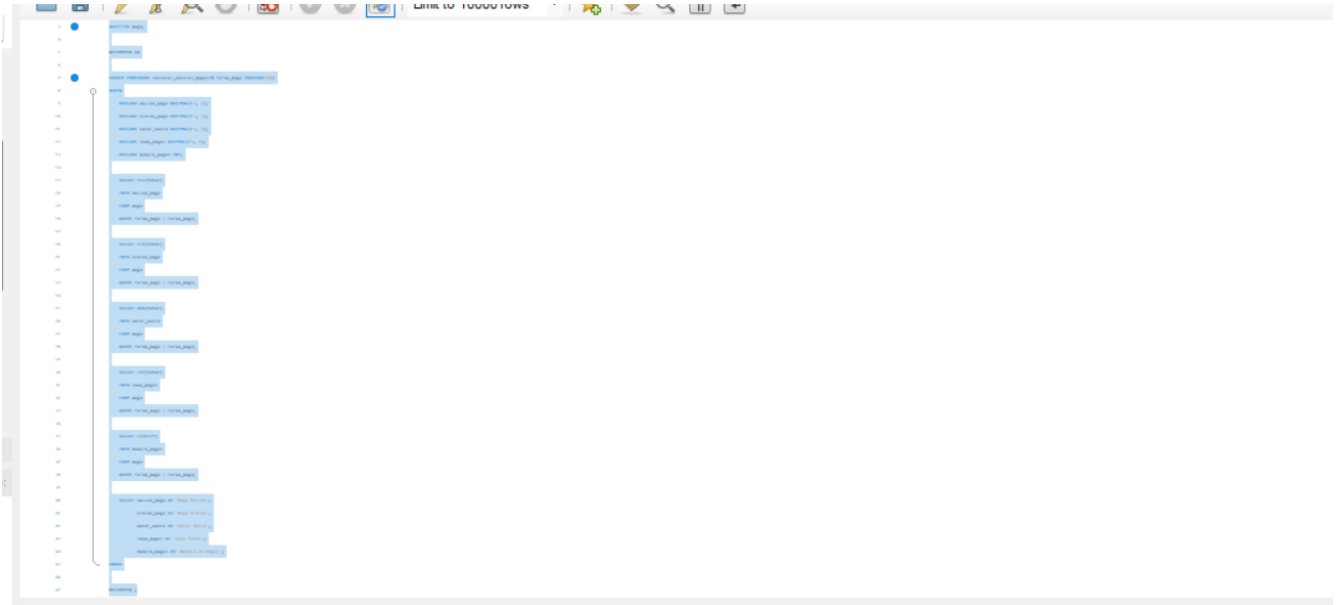
- Navigator:** Shows the 'jardineria' schema with tables like 'cliente', 'detalle_pedido', 'empleado', 'gama_producto', 'oficina', 'pago', 'pedido', 'producto', 'views', 'Stored Procedures', and 'Functions'.
- SQL Editor:** Contains the SQL code for creating the stored procedure:

```
1 use jardineria;
2 DROP PROCEDURE IF EXISTS calcular_valores_pago;
3 describe pago;
4
5 DELIMITER $$
```
- Result Grid:** Displays the structure of the 'pago' table:

Field	Type	Null	Key	Default	Extra
codigo_cliente	int	NO	PRI		
forma_pago	varchar(40)	NO			
id_transaccion	varchar(50)	NO	PRI		
fecha_pago	date	NO			
total	decimal(15,2)	NO			
- Output:** Shows the execution of the 'describe pago' command, returning 5 rows of table metadata.



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL
FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES
ACTIVIDAD PROGRAMACIÓN BASES DE DATOS



El código está demasiado largo le envío por partes

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

GF



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL
FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES
ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

```
jardineriareal*  día de la semana*  calcular valores pago* x
Limit to 10000 rows

1 • use jardineria;
2 • DROP PROCEDURE IF EXISTS calcular_valores_pago;
3 • describe pago;
4
5 DELIMITER $$
6
7 • CREATE PROCEDURE calcular_valores_pago(IN forma_pago VARCHAR(50))
8 BEGIN
9     DECLARE maximo_pago DECIMAL(15, 2);
10    DECLARE minimo_pago DECIMAL(15, 2);
11    DECLARE valor_medio DECIMAL(15, 2);
12    DECLARE suma_pagos DECIMAL(15, 2);
13    DECLARE numero_pagos INT;
14
15    SELECT MAX(total)
16    INTO maximo_pago
17    FROM pago
18    WHERE forma_pago = forma_pago;
19
20    SELECT MIN(total)
21    INTO minimo_pago
22    FROM pago
23    WHERE forma_pago = forma_pago;
24
```

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

GF



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL
FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES
ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

```
jardineriareal*   día de la semana*   calcular valores pago* x
Limit to 10000 rows

25  SELECT AVG(total)
26  INTO valor_medio
27  FROM pago
28  WHERE forma_pago = forma_pago;
29
30  SELECT SUM(total)
31  INTO suma_pagos
32  FROM pago
33  WHERE forma_pago = forma_pago;
34
35  SELECT COUNT(*)
36  INTO numero_pagos
37  FROM pago
38  WHERE forma_pago = forma_pago;
39
40  SELECT maximo_pago AS 'Pago Máximo',
41         minimo_pago AS 'Pago Mínimo',
42         valor_medio AS 'Valor Medio',
43         suma_pagos AS 'Suma Total',
44         numero_pagos AS 'Número de Pagos';
45  END$$
46
47  DELIMITER ;
48
```

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

GF



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES

ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

The screenshot shows a SQL IDE interface with a query editor, a schema browser, and a results pane. The query editor contains the following SQL code:

```
35 SELECT COUNT(*)
36 INTO numero_pagos
37 FROM pago
38 WHERE forma_pago = forma_pago;
39
40 SELECT maximo_pago AS 'Pago Máximo',
41        minimo_pago AS 'Pago Mínimo',
42        valor_medio AS 'Valor Medio',
43        suma_pagos AS 'Suma Total',
44        numero_pagos AS 'Número de Pagos';
45
46
47 DELIMITER ;
48
49 CALL calcular_valores_pago('PayPal');
50
```

The results pane displays the following data:

Pago Máximo	Pago Mínimo	Valor Medio	Suma Total	Número de Pagos
23794.00	232.00	7516.30	202940.00	27

The bottom pane shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
31	11:12:15	DROP PROCEDURE IF EXISTS calcular_valores_pago	0 row(s) affected	0.016 sec
32	11:12:15	describe pago	5 row(s) returned	0.000 sec / 0.000 sec
33	11:12:15	CREATE PROCEDURE calcular_valores_pago(IN forma_pago VARCHAR(50)) BEGIN DECLARE mas...	0 row(s) affected	0.000 sec
34	11:12:42	CALL calcular_valores_pago('PayPal')	1 row(s) returned	0.047 sec / 0.000 sec

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

GF



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES

ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

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.

The screenshot shows a MySQL IDE window with the following SQL code:

```
1 USE procedimientos;
2
3 CREATE TABLE IF NOT EXISTS pares (
4     numero INT UNSIGNED
5 );
6
7 CREATE TABLE IF NOT EXISTS impares (
8     numero INT UNSIGNED
9 );
10
11 DELIMITER $$
12
13
14 CREATE PROCEDURE calcular_pares_impares(IN tope INT UNSIGNED)
15 BEGIN
16     DECLARE contador INT DEFAULT 1;
17
18     TRUNCATE TABLE pares;
19     TRUNCATE TABLE impares;
20
21     WHILE contador <= tope DO
22         IF contador % 2 = 0 THEN
```

The Output window at the bottom shows the execution results:

#	Time	Action	Message	Duration / Fetch
32	11:12:15	describe pago	5 row(s) returned	0.000 sec / 0.000 sec
33	11:12:15	CREATE PROCEDURE calcular_valores_pago((N forma_pago VARCHAR(50)) BEGIN DECLARE maxi...	0 row(s) affected	0.000 sec
34	11:18:42	CALL calcular_valores_pago(PayPal)	1 row(s) returned	0.047 sec / 0.000 sec
35	11:21:27	USE procedimientos	0 row(s) affected	0.000 sec



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES

ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

Select * from pares

The screenshot shows the SQL Developer interface with a query window titled 'pares e impares'. The query is as follows:

```
24 ELSE
25     INSERT INTO impares (numero) VALUES (contador);
26 END IF;
27 SET contador = contador + 1;
28 END WHILE;
29 END $$
30
31 DELIMITER ;
32
33 CALL calcular_pares_impares(10);
34
35 SELECT * FROM pares;
36 SELECT * FROM impares;
37
```

The 'Result Grid' shows the output of the query, displaying the 'numero' column with values 2, 4, 6, 8, and 10. The 'Output' pane shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
41	11:23:38	SELECT * FROM impares LIMIT 0, 10000	5 row(s) returned	0.000 sec / 0.000 sec
42	11:23:51	USE procedimiento	0 row(s) affected	0.000 sec
43	11:24:00	CALL calcular_pares_impares(10)	1 row(s) affected	0.093 sec
44	11:24:06	SELECT * FROM pares LIMIT 0, 10000	5 row(s) returned	0.000 sec / 0.000 sec

Select * from impares

The screenshot shows the SQL Developer interface with a query window titled 'pares e impares'. The query is as follows:

```
24 ELSE
25     INSERT INTO impares (numero) VALUES (contador);
26 END IF;
27 SET contador = contador + 1;
28 END WHILE;
29 END $$
30
31 DELIMITER ;
32
33 CALL calcular_pares_impares(10);
34
35 SELECT * FROM pares;
36 SELECT * FROM impares;
37
```

The 'Result Grid' shows the output of the query, displaying the 'numero' column with values 1, 3, 5, 7, and 9. The 'Output' pane shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
42	11:23:51	USE procedimiento	0 row(s) affected	0.000 sec
43	11:24:00	CALL calcular_pares_impares(10)	1 row(s) affected	0.093 sec
44	11:24:06	SELECT * FROM pares LIMIT 0, 10000	5 row(s) returned	0.000 sec / 0.000 sec
45	11:26:08	SELECT * FROM impares LIMIT 0, 10000	5 row(s) returned	0.000 sec / 0.000 sec

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

GF



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES

ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

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**.

The screenshot displays the SQL Developer interface. The main editor shows the SQL code for creating the `obtener_mes` function. The function takes an integer parameter `mes` and returns a VARCHAR(20) string. It uses a `CASE` statement to map numbers 1 through 12 to the months of the year. The output window at the bottom shows the execution results of the function.

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

#	Time	Action	Message	Duration / Fetch
54	11:36:28	SELECT obtener_mes(13) AS Mes LIMIT 0, 10000	1 row(s) returned	0.000 sec / 0.000 sec
55	11:38:35	USE procedimientos	0 row(s) affected	0.000 sec
56	11:38:41	DROP FUNCTION IF EXISTS obtener_mes	0 row(s) affected	0.032 sec
57	11:38:54	CREATE FUNCTION obtener_mes(mes INT) RETURNS VARCHAR(20) DETERMINISTIC BEGIN	0 row(s) affected	0.031 sec



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES

ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

Un mes valido

The screenshot displays the SQL Developer interface with a PL/SQL function named 'obtener_mes' defined. The function uses a CASE statement to return the name of the month based on a numeric input. The execution results show the function being called with the value 13, returning 'Diciembre'.

```
21 WHEN 9 THEN SET nombre_mes = 'Septiembre';
22 WHEN 10 THEN SET nombre_mes = 'Octubre';
23 WHEN 11 THEN SET nombre_mes = 'Noviembre';
24 WHEN 12 THEN SET nombre_mes = 'Diciembre';
25 ELSE SET nombre_mes = 'Mes inválido';
26 END CASE;
27
28 RETURN nombre_mes;
29 END $$
30
31 DELIMITER ;
32
33 SELECT obtener_mes(13) AS Mes;
34
35
```

#	Time	Action	Message	Duration / Fetch
55	11:38:35	USE procedimientos	0 row(s) affected	0.000 sec
56	11:38:41	DROP FUNCTION IF EXISTS obtener_mes	0 row(s) affected	0.032 sec
57	11:38:54	CREATE FUNCTION obtener_mes(INT) RETURNS VARCHAR(20) DETERMINISTIC BEGIN	0 row(s) affected	0.031 sec
58	11:39:46	SELECT obtener_mes(13) AS Mes LIMIT 0, 10000	1 row(s) returned	0.000 sec / 0.000 sec

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

GF



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES

ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

Un mes invalido

The screenshot shows a SQL IDE interface with a PL/SQL function named `obtener_mes` defined. The function takes an integer parameter `mes` and returns a VARCHAR2 value representing the month name. The function logic is as follows:

```
WHEN 8 THEN SET nombre_mes = 'Agosto';
WHEN 9 THEN SET nombre_mes = 'Septiembre';
WHEN 10 THEN SET nombre_mes = 'Octubre';
WHEN 11 THEN SET nombre_mes = 'Noviembre';
WHEN 12 THEN SET nombre_mes = 'Diciembre';
ELSE SET nombre_mes = 'Mes inválido';
END CASE;
RETURN nombre_mes;
END $$
```

The function is then called twice in the SQL script:

```
SELECT obtener_mes(1) AS Mes;
SELECT obtener_mes(13) AS Mes;
```

The execution results are shown in the "Result Grid" and "Output" panels. The "Result Grid" shows the output of the function calls:

Mes
Mes inválido

The "Output" panel shows the execution log:

#	Time	Action	Message	Duration / Fetch
56	11:38:41	DROP FUNCTION IF EXISTS obtener_mes	0 row(s) affected	0.032 sec
57	11:38:54	CREATE FUNCTION obtener_mes(mes INT) RETURNS VARCHAR(20) DETERMINISTIC BEGIN	DEC...	0.031 sec
58	11:39:46	SELECT obtener_mes(1) AS Mes LIMIT 0, 10000	1 row(s) returned	0.000 sec / 0.000 sec
59	11:40:33	SELECT obtener_mes(13) AS Mes LIMIT 0, 10000	1 row(s) returned	0.000 sec / 0.000 sec

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

GF



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES

ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

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.

The screenshot shows a SQL IDE interface with the following components:

- Navigator:** Displays a tree view of the database schema, including tables like `empleado`, `gama_producto`, `notificaciones`, `oficina`, `pago`, `pedido`, `producto`, `Stored Procedures`, `Views`, `Functions`, `libreria`, `pizzeria`, `procedimientos`, and `Tables`.
- SQL Editor:** Contains the following SQL code:

```
1 total pagos mes año
2
3 use jardineria;
4 describe pago;
5
6 DELIMITER $$
7
8 CREATE FUNCTION obtener_total_pagos_mes_año(mes INT, año INT)
9 RETURNS DECIMAL(15,2)
10 BEGIN
```
- Result Grid:** Displays the schema of the `pago` table:

Field	Type	Null	Key	Default	Extra
codigo_cliente	int	NO	PRI		
forma_pago	varchar(40)	NO			
id_transaccion	varchar(50)	NO	PRI		
fecha_pago	date	NO			
total	decimal(15,2)	NO			
- Action Output:** Displays the execution results of the queries:

#	Time	Action	Message	Duration / Fetch
58	11:39:46	SELECT obtener_mes(1) AS Mes LIMIT 0, 10000	1 row(s) returned	0.000 sec / 0.000 sec
59	11:40:33	SELECT obtener_mes(13) AS Mes LIMIT 0, 10000	1 row(s) returned	0.000 sec / 0.000 sec
60	11:42:24	use jardineria	0 row(s) affected	0.015 sec
61	11:42:24	describe pago	5 row(s) returned	0.000 sec / 0.000 sec

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

GF



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES

ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

The screenshot displays the SQL Server Enterprise Manager interface. The left pane shows the 'SCHEMAS' tree with the 'jardineria' database selected. The right pane shows the SQL script editor with the following code:

```
use jardineria;
describe pago;

DELIMITER $$
CREATE FUNCTION obtener_total_pagos_mes_año(mes INT, año INT)
RETURNS DECIMAL(15,2)
DETERMINISTIC
BEGIN
    DECLARE total_pagos DECIMAL(15,2);
    SELECT SUM(total)
    INTO total_pagos
    FROM pago
    WHERE MONTH(fecha_pago) = mes
    AND YEAR(fecha_pago) = año;
    RETURN IFNULL(total_pagos, 0);
END$$
DELIMITER ;
```

The 'Output' pane shows the execution results:

#	Time	Action	Message	Duration / Fetch
70	11:45:14	DROP FUNCTION IF EXISTS obtener_total_pagos_mes_año	0 row(s) affected	0.015 sec
71	11:45:22	use jardineria	0 row(s) affected	0.000 sec
72	11:45:26	describe pago	5 row(s) returned	0.000 sec / 0.000 sec
73	11:45:33	CREATE FUNCTION obtener_total_pagos_mes_año(mes INT, año INT) RETURNS DECIMAL(15,2)	0 row(s) affected	0.032 sec

The 'Result Grid' shows the output of the function call:

total_pagos_enero_2009
26752.00

The 'Output' pane also shows the execution results for the function call:

#	Time	Action	Message	Duration / Fetch
71	11:45:22	use jardineria	0 row(s) affected	0.000 sec
72	11:45:26	describe pago	5 row(s) returned	0.000 sec / 0.000 sec
73	11:45:33	CREATE FUNCTION obtener_total_pagos_mes_año(mes INT, año INT) RETURNS DECIMAL(15,2)	0 row(s) affected	0.032 sec
74	11:45:04	SELECT obtener_total_pagos_mes_año(1, 2009) AS total_pagos_enero_2009 LIMIT 0, 10000	1 row(s) returned	0.016 sec / 0.000 sec

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

GF



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES

ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

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.

Describe productos

The screenshot shows the SQL Developer interface with the following components:

- Navigator:** Shows the 'jardineria' schema with tables like 'empleados', 'gama_producto', 'notificaciones', 'oficina', 'pago', 'pedido', 'productos', 'Views', 'Stored Procedures', 'Functions', 'libreria', 'pizzeria', 'procedimientos', and 'Tables'.
- SQL Editor:** Contains the following SQL code:

```
1 use jardineria;
2 DROP FUNCTION IF EXISTS cantidad_total_de_productos_vendidos;
3 describe producto;
4
5 DELIMITER $$
6
7 CREATE FUNCTION cantidad_total_de_productos_vendidos(codigo_producto VARCHAR(50))
8 RETURNS INT
9 DETERMINISTIC
10 BEGIN
11     DECLARE total_productos INT;
12
13     -- Calcular la cantidad total de productos vendidos con el código especificado
14     SELECT SUM(cantidad)
```
- Result Grid:** Displays the schema of the 'productos' table:

Field	Type	Null	Key	Default	Extra
codigo_producto	varchar(15)	NO	PRI		
nombre	varchar(70)	NO			
gama	varchar(50)	NO	MUL		
dimensiones	varchar(25)	YES			
proveedor	varchar(50)	YES			
descripcion	text	YES			
- Output:** Shows the execution results of the SQL statements:

#	Time	Action	Message	Duration / Fetch
74	11:46:04	SELECT obtener_total_pagos_mes_año(1, 2009) AS total_pagos_enero_2009 LIMIT 0, 10000	1 row(s) returned	0.016 sec / 0.000 sec
75	12:14:04	use jardineria	0 row(s) affected	0.000 sec
76	12:14:08	DROP FUNCTION IF EXISTS cantidad_total_de_productos_vendidos	0 row(s) affected	0.015 sec
77	12:14:13	describe producto	9 row(s) returned	0.000 sec / 0.000 sec

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

GF



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES

ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Schemas' tree with 'jardineria' selected. The central pane shows a SQL script for creating a function. The right pane displays the 'Output' window with the execution results.

```
4
5
6
7 CREATE FUNCTION cantidad_total_de_productos_vendidos(codigo_producto VARCHAR(50))
8 RETURNS INT
9 DETERMINISTIC
10 BEGIN
11     DECLARE total_productos INT;
12     -- Calcular la cantidad total de productos vendidos con el código especificado
13     SELECT SUM(cantidad)
14     INTO total_productos
15     FROM detalle_pedido
16     WHERE codigo_producto = codigo_producto;
17     -- Devolver el total calculado
18     RETURN IFNULL(total_productos, 0);
19 END;
20
21
22
23
24
25 SELECT cantidad_total_de_productos_vendidos('PR-67') AS total_vendido;
```

#	Time	Action	Message	Duration / Fetch
75	12:14:04	use jardineria	0 row(s) affected	0.000 sec
76	12:14:08	DROP FUNCTION IF EXISTS cantidad_total_de_productos_vendidos	0 row(s) affected	0.015 sec
77	12:14:13	describe producto	9 row(s) returned	0.000 sec / 0.000 sec
78	12:14:51	CREATE FUNCTION cantidad_total_de_productos_vendidos(codigo_producto VARCHAR(50)) RET...	0 row(s) affected	0.032 sec

The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the 'Schemas' tree with 'jardineria' selected. The central pane shows a SQL script for creating a function. The right pane displays the 'Output' window with the execution results.

```
15
16
17
18
19
20
21
22
23
24
25
26
27
28
```

#	Time	Action	Message	Duration / Fetch
76	12:14:08	DROP FUNCTION IF EXISTS cantidad_total_de_productos_vendidos	0 row(s) affected	0.015 sec
77	12:14:13	describe producto	9 row(s) returned	0.000 sec / 0.000 sec
78	12:14:51	CREATE FUNCTION cantidad_total_de_productos_vendidos(codigo_producto VARCHAR(50)) RET...	0 row(s) affected	0.032 sec
79	12:15:17	SELECT cantidad_total_de_productos_vendidos('PR-67') AS total_vendido LIMIT 0, 10000	1 row(s) returned	0.047 sec / 0.000 sec

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

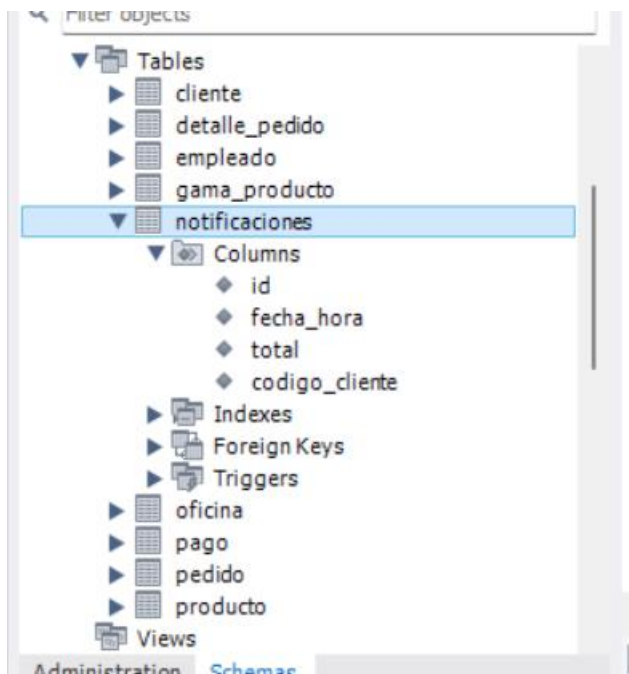
GF



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL
FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES
ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

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)





PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES

ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

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.

The screenshot shows a database management tool interface. The main window displays SQL code for creating a trigger named `trigger_notificar_pago`. The code is as follows:

```
14
15 CREATE TRIGGER trigger_notificar_pago
16 AFTER INSERT ON pago
17 FOR EACH ROW
18 BEGIN
19     INSERT INTO notificaciones (fecha_hora, total, codigo_cliente)
20     VALUES (NEW.fecha_pago, NEW.total, NEW.codigo_cliente);
21 END$$
22
23 DELIMITER ;
24
25 INSERT INTO pago (codigo_cliente, forma_pago, id_transaccion, fecha_pago, total)
26 VALUES (1, 'PayPal', 'txn001', NOW(), 5000);
27
```

Below the code, the 'Result Grid' shows the execution of the SQL statements. The first statement is the trigger creation, and the second is the data insertion. The 'Output' pane shows the execution log with the following details:

#	Time	Action	Message	Duration / Fetch
122	12:39:02	use jardineria	0 row(s) affected	0.000 sec
123	12:39:03	use jardineria	0 row(s) affected	0.000 sec
124	12:39:07	show tables	9 row(s) returned	0.015 sec / 0.000 sec
125	12:39:27	SELECT * FROM notificaciones LIMIT 0, 10000	1 row(s) returned	0.000 sec / 0.000 sec

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

GF



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL
FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES
ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

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)

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)

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL
FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES
ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

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

Output			OUT email VARCHAR(100)	
Action Output)	
#	Time	Action	BEGIN	
973	12:49:48	CREATE TABLE alumnos (id INT UNSIGNED	SET email = LOWER(CONCAT(
974	12:49:58	CREATE PROCEDURE crear_email(IN no	LEFT(nombre, 1),	
975	12:50:04	CALL crear_email('Juan', 'Perez', 'Lopez', 'test	LEFT(apellido1, 3),	
976	12:50:10	SELECT @resultado LIMIT 0, 10000	LEFT(apellido2, 3),	
			'@',	
			dominio	
));	
			END	
			Message	Duration / Fetch
			?, nombre V... 0 row(s) affected	0.046 sec
			RCHAR(50), ... 0 row(s) affected	0.032 sec
			0 row(s) affected	0.000 sec
			1 row(s) returned	0.000 sec / 0.000 sec

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

GF



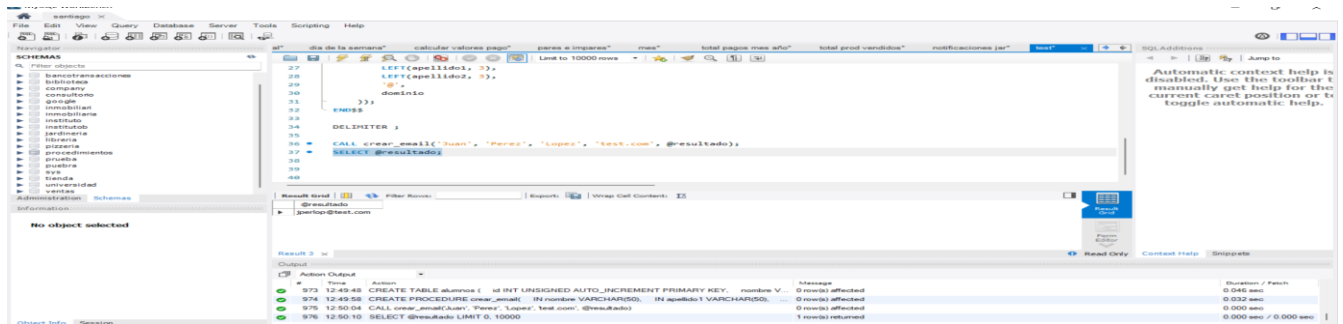
PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL

FORMATO GUÍA DE APRENDIZAJE – ACTIVIDADES

ACTIVIDAD PROGRAMACIÓN BASES DE DATOS

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.



Nota: Para crear la nueva dirección de email se deberá hacer uso del procedimiento **crear_email**.

Fecha límite entrega evidencias: diciembre 5 de 2024

Se debe entregar como evidencia el documento PDF con la solución y la base de datos. Les recuerdo imprimir los pantallazos de cada solución que incluye la ejecución a la hora de crearlos y la prueba de cada uno de ellos.

CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	César Marino Cuéllar Chacón	Instructor	CTPI-CAUCA	26-11-2024

Tecnólogo en Análisis y Desarrollo de Software Servicio Nacional de Aprendizaje SENA
Centro de Teleinformática y Producción Industrial Regional Cauca

Popayán, día 05 de mes diciembre del año 2024

GF