

Taller de procedimientos almacenados

Karen Michel Palacios López

Procedimientos almacenados

Profesor: Ing. Mary Luz Rubiano Acosta



Universidad de San Buenaventura

Facultad de ingeniería

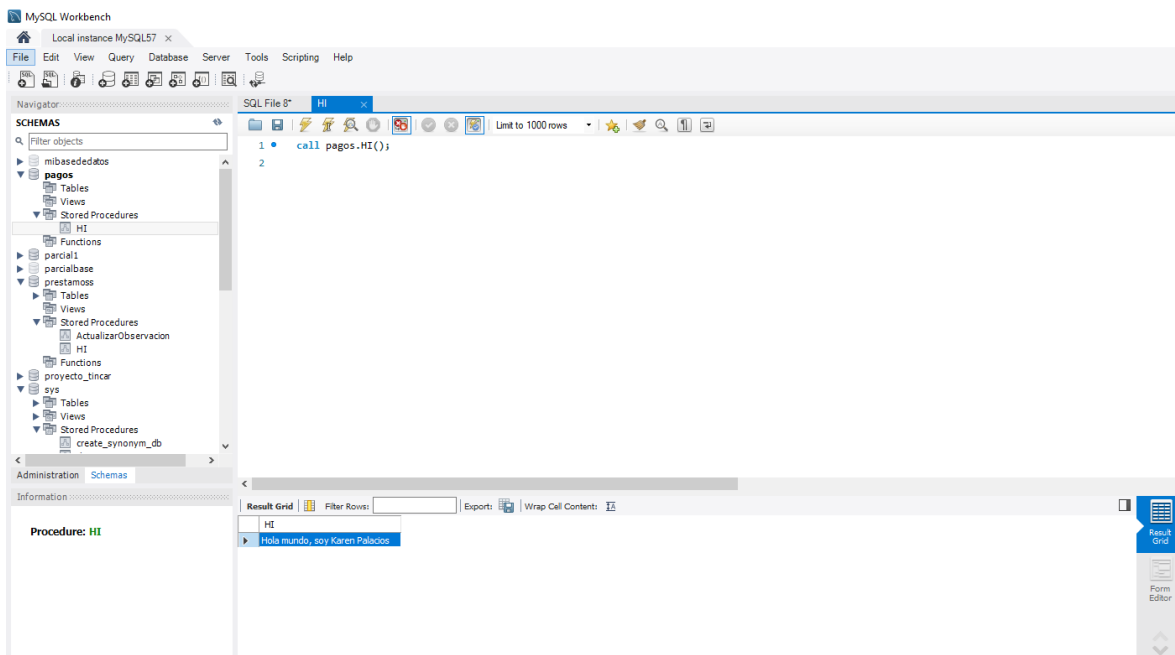
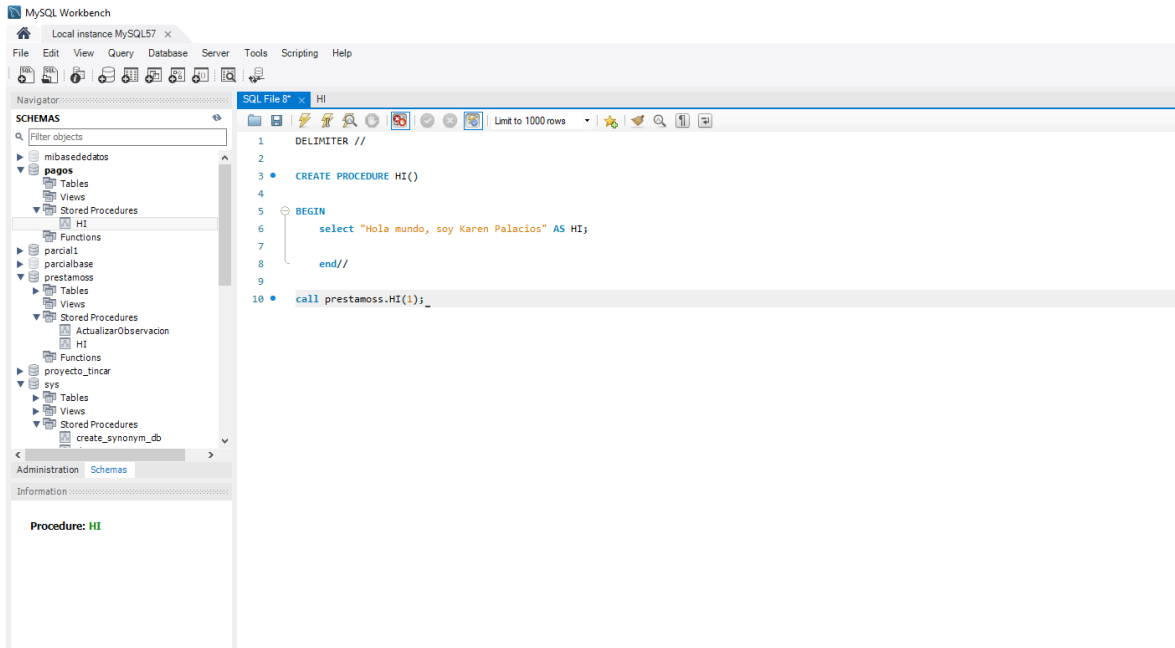
Tecnología en desarrollo de software

Bogotá D.C.

2024

Taller de procedimientos almacenados

1. Escribe un procedimiento que no tenga ningún parámetro de entrada ni de salida y que muestre el texto ¡Hola mundo!



2. Escribe 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

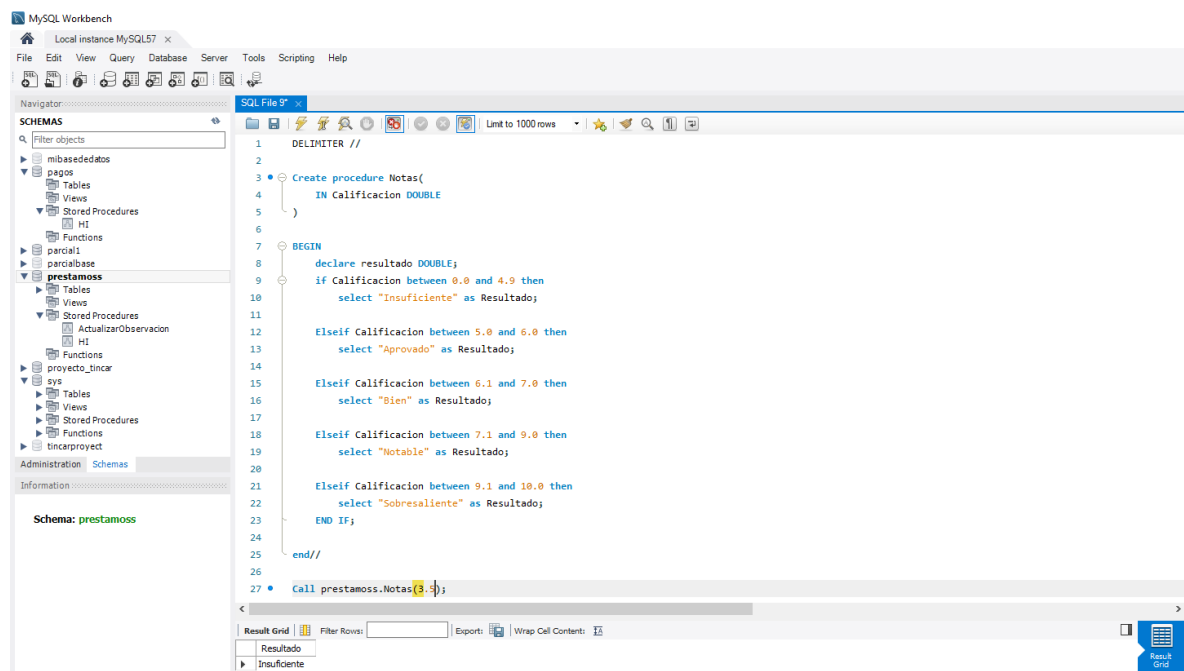
[5,6) = Aprobado

[6, 7) = Bien

[7, 9) = Notable

[9, 10] = Sobresaliente

En cualquier otro caso la nota no será válida.



Navigator: SCHEMAS

Filter objects

- mibasedatos
 - pagos
 - Tables
 - Views
 - Stored Procedures
 - HI
 - Functions
 - parcial1
 - parcialbase
 - prestamos
 - Tables
 - Views
 - Stored Procedures
 - ActualizarObservacion
 - HI
 - Functions
 - proyecto_tincar
 - sys
 - Tables
 - Views
 - Stored Procedures
 - Functions
 - tincaproject

Administration Schemas

Information

Schema: prestamos

SQL File 9'

```
1 DELIMITER //
```

```
2
```

```
3 Create procedure Notas(
```

```
4     IN Calificacion DOUBLE
```

```
5 )
```

```
6
```

```
7 BEGIN
```

```
8     declare resultado DOUBLE;
```

```
9     if Calificacion between 0.0 and 4.9 then
```

```
10         select "Insuficiente" as Resultado;
```

```
11
```

```
12     ElseIf Calificacion between 5.0 and 6.0 then
```

```
13         select "Aprovado" as Resultado;
```

```
14
```

```
15     ElseIf Calificacion between 6.1 and 7.0 then
```

```
16         select "Bien" as Resultado;
```

```
17
```

```
18     ElseIf Calificacion between 7.1 and 9.0 then
```

```
19         select "Notable" as Resultado;
```

```
20
```

```
21     ElseIf Calificacion between 9.1 and 10.0 then
```

```
22         select "Sobresaliente" as Resultado;
```

```
23     END IF;
```

```
24
```

```
25 end//
```

```
26
```

```
27 Call prestamos.Notas(5.8);
```

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

Resultado
Aprovado

Result Grid
Form Editor

Local instance MySQL57 x

File Edit View Query Database Server Tools Scripting Help

Navigator: SCHEMAS

Filter objects

- mibasedatos
 - pagos
 - Tables
 - Views
 - Stored Procedures
 - HI
 - Functions
 - parcial1
 - parcialbase
 - prestamos
 - Tables
 - Views
 - Stored Procedures
 - ActualizarObservacion
 - HI
 - Functions
 - proyecto_tincar
 - sys
 - Tables
 - Views
 - Stored Procedures
 - Functions
 - tincaproject

Administration Schemas

Information

Schema: prestamos

SQL File 9'

```
1 DELIMITER //
```

```
2
```

```
3 Create procedure Notas(
```

```
4     IN Calificacion DOUBLE
```

```
5 )
```

```
6
```

```
7 BEGIN
```

```
8     declare resultado DOUBLE;
```

```
9     if Calificacion between 0.0 and 4.9 then
```

```
10         select "Insuficiente" as Resultado;
```

```
11
```

```
12     ElseIf Calificacion between 5.0 and 6.0 then
```

```
13         select "Aprovado" as Resultado;
```

```
14
```

```
15     ElseIf Calificacion between 6.1 and 7.0 then
```

```
16         select "Bien" as Resultado;
```

```
17
```

```
18     ElseIf Calificacion between 7.1 and 9.0 then
```

```
19         select "Notable" as Resultado;
```

```
20
```

```
21     ElseIf Calificacion between 9.1 and 10.0 then
```

```
22         select "Sobresaliente" as Resultado;
```

```
23     END IF;
```

```
24
```

```
25 end//
```

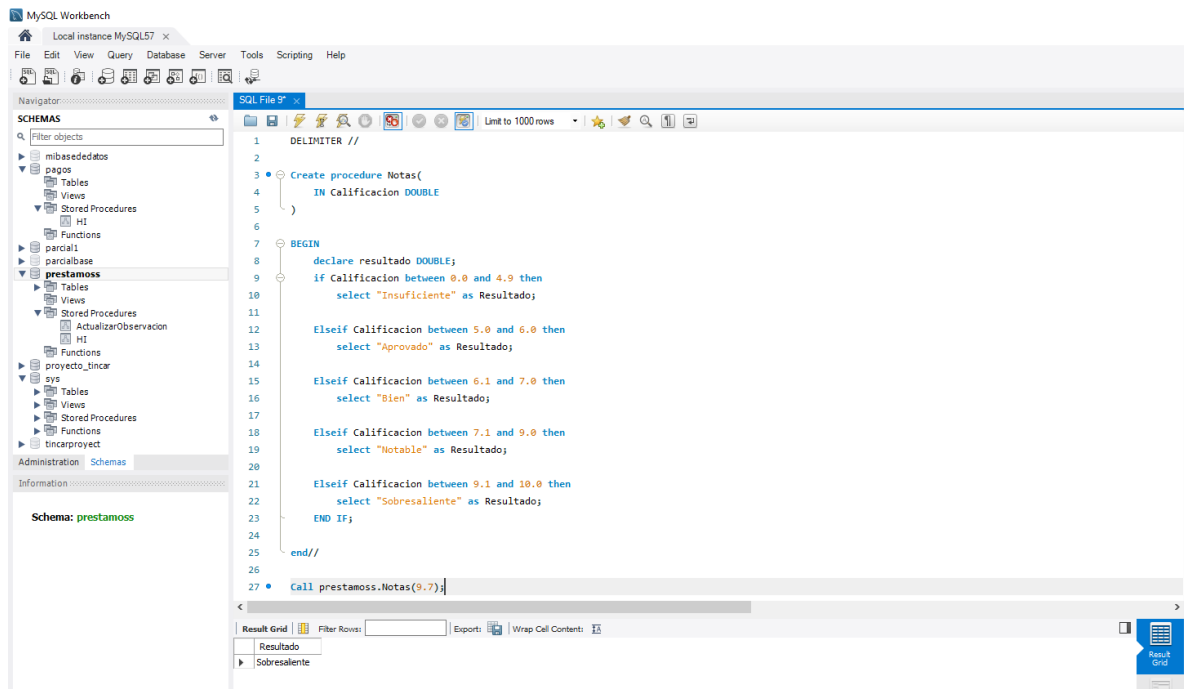
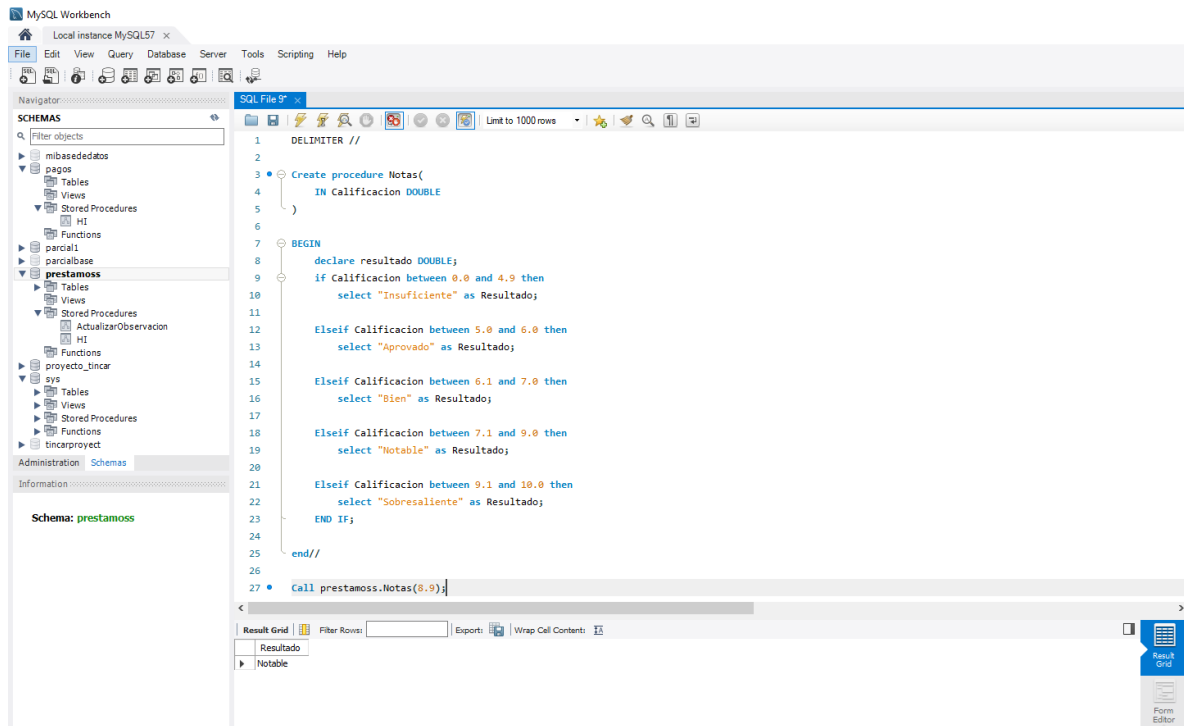
```
26
```

```
27 Call prestamos.Notas(6.3);
```

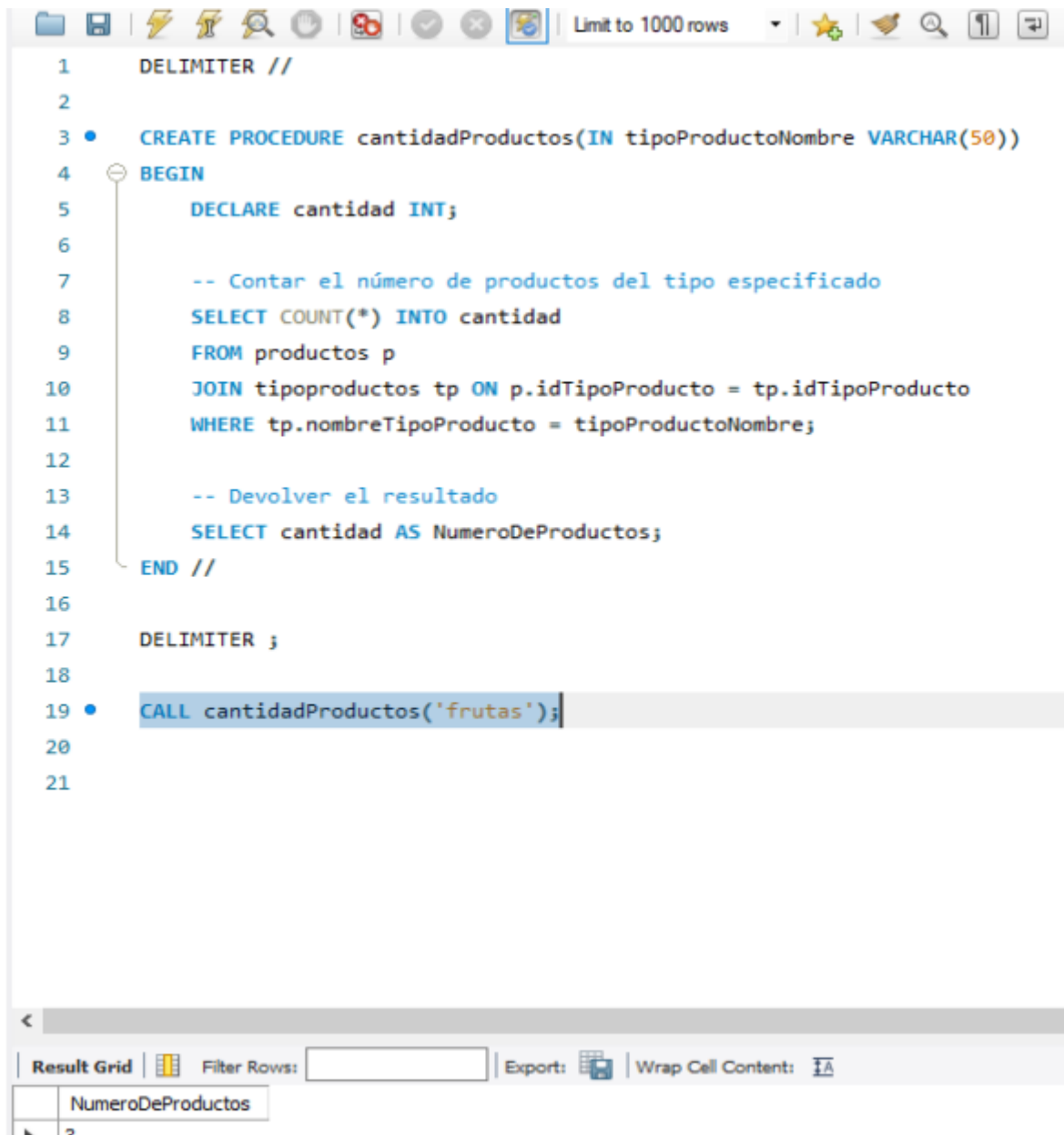
Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Resultado
Bien

Result Grid
Form Editor



3. Escriba un procedimiento llamado cantidad Productos que reciba como entrada el nombre del tipo de producto y devuelva el número de productos que existen dentro de esa categoría.




The screenshot shows a SQL IDE interface. The main editor contains the following SQL code:

```
1 DELIMITER //
2
3 CREATE PROCEDURE cantidadProductos(IN tipoProductoNombre VARCHAR(50))
4 BEGIN
5     DECLARE cantidad INT;
6
7     -- Contar el número de productos del tipo especificado
8     SELECT COUNT(*) INTO cantidad
9     FROM productos p
10    JOIN tipoproductos tp ON p.idTipoProducto = tp.idTipoProducto
11    WHERE tp.nombreTipoProducto = tipoProductoNombre;
12
13    -- Devolver el resultado
14    SELECT cantidad AS NumeroDeProductos;
15 END //
16
17 DELIMITER ;
18
19 CALL cantidadProductos('frutas');
```

The code is numbered 1 through 21. Line 19 is highlighted. The IDE has a toolbar at the top with icons for file operations, execution, and search. A status bar at the top right indicates "Limit to 1000 rows". At the bottom, there is a "Result Grid" section with a "Filter Rows:" input field, an "Export:" button, and a "Wrap Cell Content:" checkbox. Below this, a table is visible with one column labeled "NumeroDeProductos" and one row containing the value "3".

4. Escribe un procedimiento que se llame `preciosProductos`, que reciba como parámetro de entrada el nombre del tipo de producto y devuelva como salida tres parámetros. El precio máximo, el precio mínimo y la media de los productos que existen en esa categoría.



Limit to 1000 rows




```
1 DELIMITER //
```

```
2
```

```
3 CREATE PROCEDURE preciosProductos(  
4     IN tipoProductoNombre VARCHAR(50),  
5     OUT precioMaximo DECIMAL(10, 2),  
6     OUT precioMinimo DECIMAL(10, 2),  
7     OUT precioPromedio DECIMAL(10, 2)  
8 )  
9 BEGIN  
10  
11     SET precioMaximo = 0;  
12     SET precioMinimo = 0;  
13     SET precioPromedio = 0;  
14  
15  
16     SELECT  
17         MAX(valorVenta) AS maxPrecio,  
18         MIN(valorVenta) AS minPrecio,  
19         AVG(valorVenta) AS avgPrecio  
20     INTO precioMaximo, precioMinimo, precioPromedio  
21     FROM productos p  
22     JOIN tipoproductos tp ON p.idTipoProducto = tp.idTipoProducto  
23     WHERE tp.nombreTipoProducto = tipoProductoNombre;  
--
```

```
27 DELIMITER ;
28 • SET @maxPrecio = 0;
29 • SET @minPrecio = 0;
30 • SET @promedioPrecio = 0;
31 |
32 • CALL preciosProductos('frutas', @maxPrecio, @minPrecio, @promedioPrecio);
33
34 • SELECT
35     @maxPrecio AS PrecioMaximo,
36     @minPrecio AS PrecioMinimo,
37     @promedioPrecio AS PrecioPromedio;
38
39
40
```

<

Result Grid  Filter Rows: Export:  Wrap Cell Content: 

	PrecioMaximo	PrecioMinimo	PrecioPromedio
▶	2000.00	1000.00	1400.00

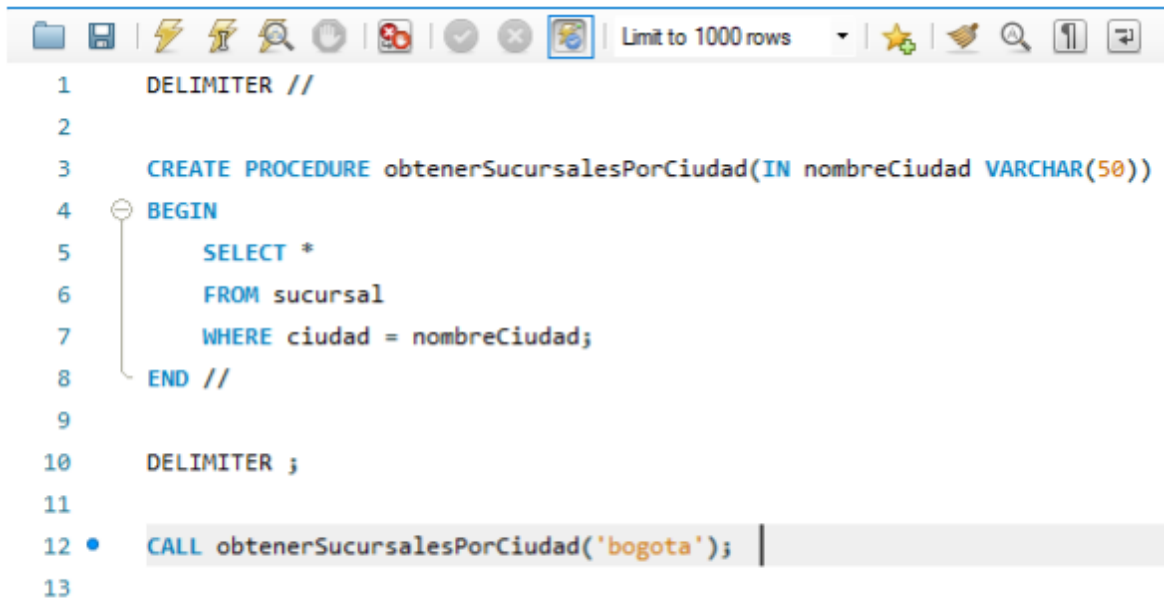
5. Realice un procedimiento que se llame funcionIVA que incluya una función que calcule el total con el incremento del IVA.


```
1 DELIMITER //
2
3 CREATE FUNCTION calcularIVA(precio DECIMAL(10, 2))
4 RETURNS DECIMAL(10, 2)
5 DETERMINISTIC
6 BEGIN
7     DECLARE ivaRate DECIMAL(5, 2) DEFAULT 19;
8     RETURN precio + (precio * ivaRate / 100);
9 END //
10
11 DELIMITER ;
12 DELIMITER //
13
14 CREATE PROCEDURE funcionIVA(
15     IN precioBase DECIMAL(10, 2),
16     OUT totalConIVA DECIMAL(10, 2)
17 )
18 BEGIN
19     SET totalConIVA = calcularIVA(precioBase);
20 END //
21
22 DELIMITER ;
23
24 SET @totalConIVA = 0;
25
26 CALL funcionIVA(100, @totalConIVA);
27
28 SELECT @totalConIVA AS TotalConIVA;
29
30
```

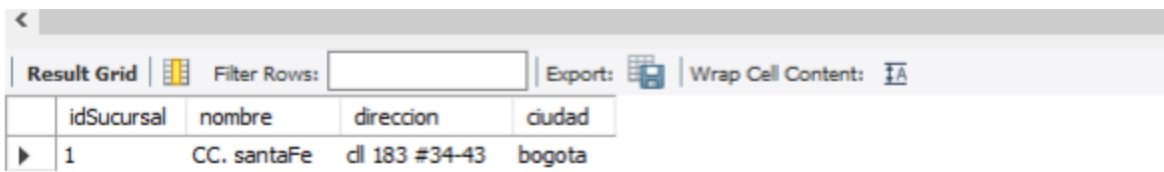
Result Grid

TotalConIVA
119.00

6. Escribe un procedimiento que reciba el nombre de un país como parámetro de entrada y realice una consulta sobre la tabla sucursal para obtener todos las sucursales que existen en la tabla de ese país.

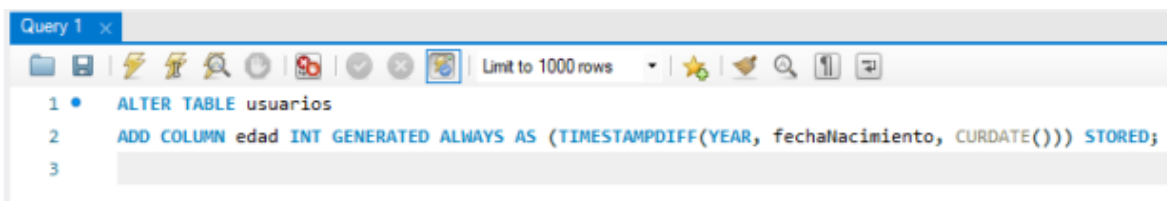


```
1 DELIMITER //
2
3 CREATE PROCEDURE obtenerSucursalesPorCiudad(IN nombreCiudad VARCHAR(50))
4 BEGIN
5     SELECT *
6     FROM sucursal
7     WHERE ciudad = nombreCiudad;
8 END //
9
10 DELIMITER ;
11
12 CALL obtenerSucursalesPorCiudad('bogota');
```



	idSucursal	nombre	direccion	ciudad
1		CC. santaFe	cl 183 #34-43	bogota

7. Una vez creada la tabla se decide añadir una nueva columna a la tabla llamada edad que será un valor calculado a partir de la columna fecha_nacimiento. Escriba la sentencia SQL necesaria para modificar la tabla y añadir la nueva columna.



```
1 ALTER TABLE usuarios
2 ADD COLUMN edad INT GENERATED ALWAYS AS (TIMESTAMPDIFF(YEAR, fechaNacimiento, CURDATE())) STORED;
3
```

8. Escriba una función llamada calcularEdad que reciba una fecha y devuelva el número de años que han pasado desde la fecha actual hasta la fecha pasada como parámetro:

The screenshot shows a SQL query editor window titled "Query 1". The query is as follows:

```
1 DELIMITER //  
2  
3 CREATE FUNCTION calcularEdad(fechaNacimiento DATE)  
4 RETURNS INT  
5 DETERMINISTIC  
6 BEGIN  
7     DECLARE edad INT;  
8  
9     SET edad = TIMESTAMPDIFF(YEAR, fechaNacimiento, CURDATE());  
10  
11     RETURN edad;  
12 END //  
13  
14 DELIMITER ;  
15  
16 • SELECT calcularEdad('2000-01-01') AS edad;  
17
```

Below the query editor, the "Result Grid" is visible, showing the output of the query:

edad
24

9. Escriba un procedimiento que permita calcular la edad de todos los usuarios que ya existen en la tabla. Para esto será necesario crear un procedimiento llamado `actualizarColumnaEdad` que calcule la edad de cada usuario y actualice la tabla. Este procedimiento hará uso de la función `calcularEdad` que hemos creado en el paso anterior.

Query 1 x









Limit to 1000 rows

```
1
2 DELIMITER //
3
4 • CREATE PROCEDURE actualizarColumnaEdad()
5 BEGIN
6     DECLARE usuario_id INT;
7     DECLARE fecha_nacimiento DATE;
8     DECLARE edad_calculada INT;
9
10    DECLARE cur CURSOR FOR
11    SELECT idUsuarios, fechaNacimiento FROM usuarios;
12
13    DECLARE CONTINUE HANDLER FOR NOT FOUND SET usuario_id = NULL;
14
15    OPEN cur;
16
17
18    read_loop: LOOP
19        FETCH cur INTO usuario_id, fecha_nacimiento;
20
21
22        IF usuario_id IS NULL THEN
23            LEAVE read_loop;
24        END IF;
```

```

27         SET edad_calculada = calcularEdad(fecha_nacimiento);
28
29
30         UPDATE usuarios
31         SET edad = edad_calculada
32         WHERE idUsuarios = usuario_id;
33     END LOOP;
34
35     CLOSE cur;
36 END //
37
38 DELIMITER ;
39 • CALL actualizarColumnaEdad();
40 • SELECT * FROM usuarios;
41
42
43

```

Result Grid					
Filter Rows: <input type="text"/>					
Edit:    Export/Import:   Wrap Cell Content: 					
	idUsuarios	nombre	apellido	fechaNacimiento	edad
▶	1	Juan	Pérez	1990-01-15	34
	2	Ana	Gómez	1985-05-30	39
	3	Luis	Martínez	2000-09-20	24
*	NULL	NULL	NULL	NULL	NULL

10. Escribe un procedimiento almacenado para su proyecto integrador que sea útil.
Tercer punto “Cantidad de productos”

Query 1 autos

Limit to 1000 rows

```
1 DELIMITER //
2
3 CREATE PROCEDURE insertarCarro(
4     IN p_marca VARCHAR(50),
5     IN p_color VARCHAR(30),
6     IN p_modelo VARCHAR(50),
7     IN p_matricula VARCHAR(20)
8 )
9 BEGIN
10     INSERT INTO carros (marca, color, modelo, matricula)
11     VALUES (p_marca, p_color, p_modelo, p_matricula);
12 END //
13
14 DELIMITER ;
15
16 CALL insertarCarro('Toyota', 'Rojo', 'Corolla', 'ABC123');
17 CALL insertarCarro('Honda', 'Azul', 'Civic', 'XYZ789');
18
19 SELECT * FROM carros;
20
```

Result Grid

Filter Rows:

Edit: Export/Import: Wrap Cell

	id	marca	color	modelo	matricula
▶	1	Toyota	Rojo	Corolla	ABC123
	2	Honda	Azul	Civic	XYZ789
*		NULL	NULL	NULL	NULL