

Universidad de San Buenaventura

Facultad de Ingeniería

Programa Desarrollo de Software

Materia: Programación de Bases de Datos

Trabajo:

Procedimientos Almacenados

Presentado por:

José Daniel Silva Berrio

Leidy Johana Sarmiento

Profesora:

Mary Luz Rubiano

**Bogotá, Colombia
2024**

TALLER PROCEDIMIENTOS

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

The image shows two screenshots of a SQL Studio interface. The top screenshot displays the creation of a stored procedure named 'mostrarHolaMundo' in the 'modava' schema. The DDL editor shows the following code:

```
1 CREATE DEFINER='root'@'localhost' PROCEDURE `mostrarHolaMundo`()
2 BEGIN
3     SELECT '¡Hola mundo!';
4 END
```

The bottom screenshot shows the execution of the procedure. The SQL editor contains the command:

```
1 call modava.mostrarHolaMundo();
```

The 'Result Grid' at the bottom displays the output of the procedure:

¡Hola mundo!

The 'Schema: modava' is indicated at the bottom left of the interface.

Ejercicio 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

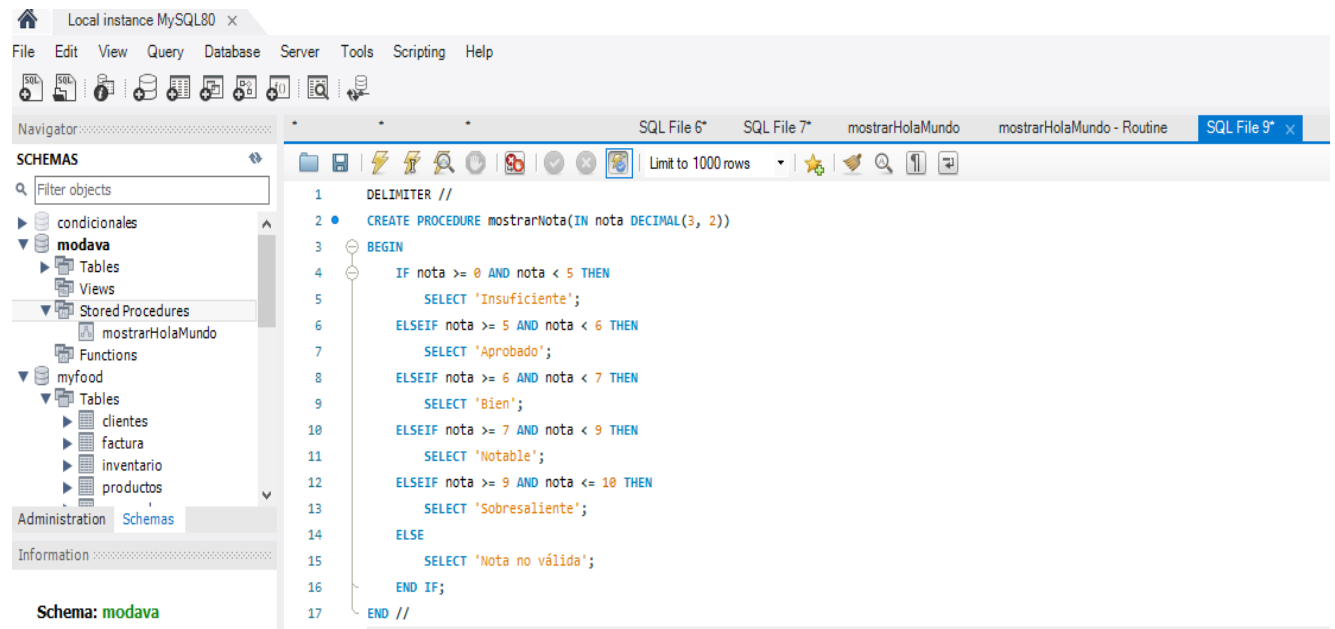
[5,6) = Aprobado

[6, 7) = Bien

[7, 9) = Notable

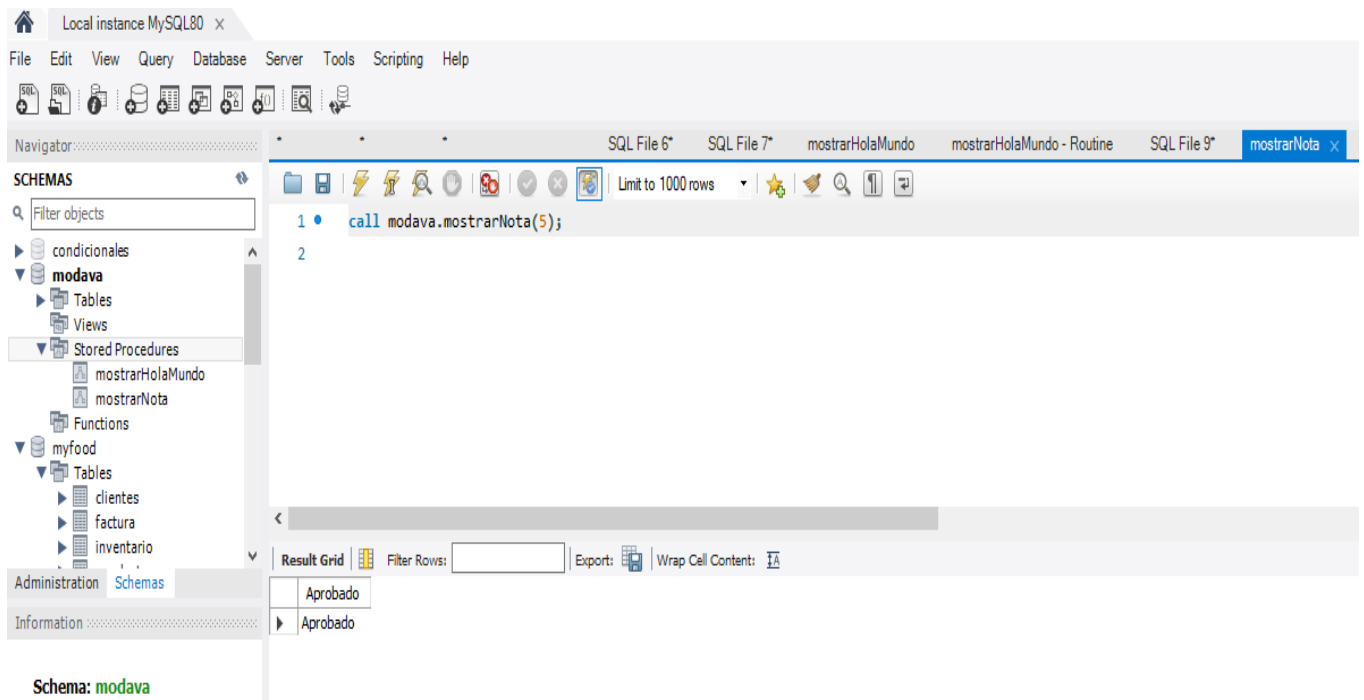
[9, 10] = Sobresaliente.

En cualquier otro caso la nota no será válida.

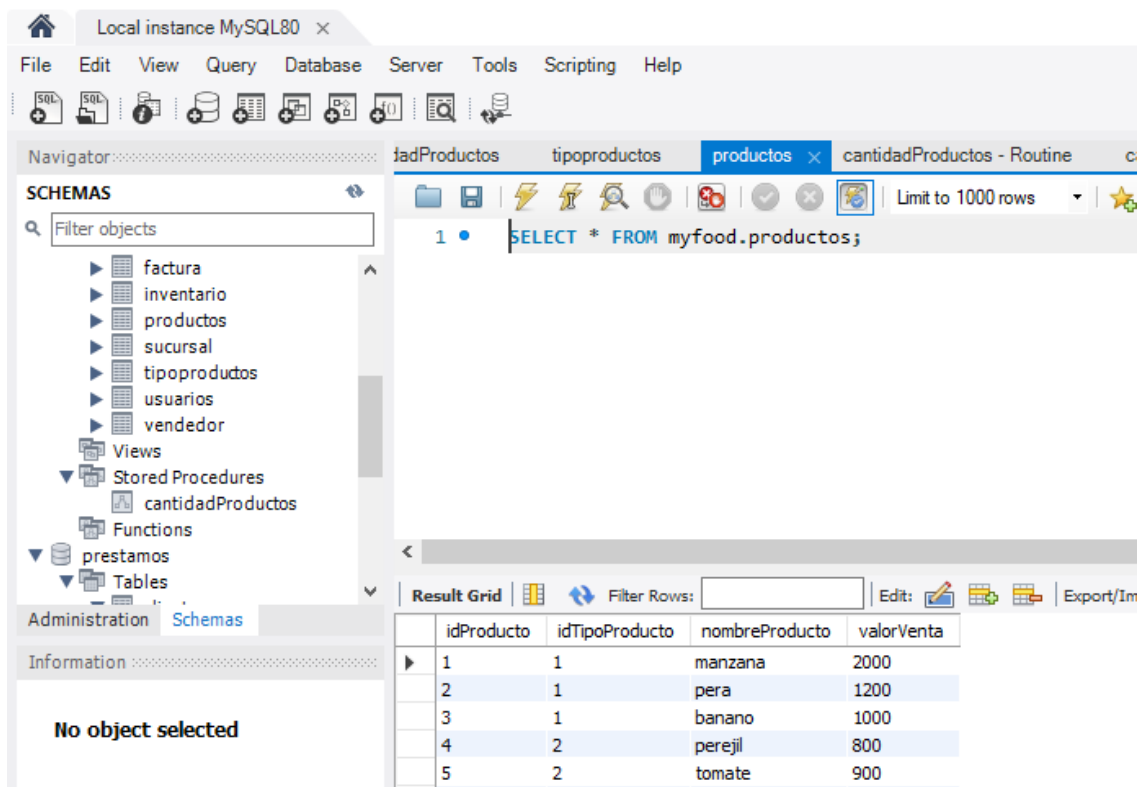


The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a tree view with 'modava' selected. The main editor displays the following SQL code:

```
1 DELIMITER //
2 CREATE PROCEDURE mostrarNota(IN nota DECIMAL(3, 2))
3 BEGIN
4     IF nota >= 0 AND nota < 5 THEN
5         SELECT 'Insuficiente';
6     ELSEIF nota >= 5 AND nota < 6 THEN
7         SELECT 'Aprobado';
8     ELSEIF nota >= 6 AND nota < 7 THEN
9         SELECT 'Bien';
10    ELSEIF nota >= 7 AND nota < 9 THEN
11        SELECT 'Notable';
12    ELSEIF nota >= 9 AND nota <= 10 THEN
13        SELECT 'Sobresaliente';
14    ELSE
15        SELECT 'Nota no válida';
16    END IF;
17 END //
```



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



Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: cantidadProductos - Routine cantidadProductos - Routine cantidadProductos cantidadProductos - Routine

SCHEMAS

Filter objects

Functions

myfood

Tables

- clientes
- factura
- inventario
- productos
- sucursal
- tipoproductos
- usuarios
- vendedor

Views

Stored Procedures

Administration Schemas

Information

1 • select count(productos.idTipoproducto) FROM productos

2 join tipoproductos

3 ON productos.idTipoProducto = tipoproductos.idTipoProducto

4 where tipoproductos.nombreTipoProducto = "frutas"

5 GROUP BY productos.idtipoproducto;

Result Grid

count(productos.idTipoproducto)
3

MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator: cantidadProductos - Routine cantidadProductos - Routine cantidadProductos cantidadProductos - Routine Si

SCHEMAS

Filter objects

Functions

myfood

Tables

- clientes
- factura
- inventario
- productos
- sucursal
- tipoproductos
- usuarios
- vendedor

Views

Stored Procedures

Administration Schemas

Information

1

2 • call myfood.cantidadProductos('frutas');

3

Result Grid

count(productos.idTipoproducto)
3

No object selected

Result 1 x

Output

Action Output

#	Time	Action	Message
✓ 4	20:23:01	DROP PROCEDURE 'myfood'. 'cantidadProductos'	0 row(s) affected
✓ 5	20:23:05	Apply changes to cantidadProductos	Changes applied
✓ 6	20:23:29	call myfood.cantidadProductos(frutas)	0 row(s) returned
✓ 7	20:24:20	select count(productos.idTipoproducto) FROM productos join tipoproductos ON producto...	1 row(s) returned
✓ 8	20:24:51	Apply changes to cantidadProductos	Changes applied
✓ 9	20:25:05	call myfood.cantidadProductos(frutas)	1 row(s) returned

Object Info Session

Ejercicio 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.

The screenshot shows the SQL Studio interface. On the left, the 'SCHEMAS' pane shows a tree view of the database structure, including 'factura', 'inventario', 'productos', 'sucursal', 'tipoproductos', 'usuarios', 'vendedor', 'Views', 'Stored Procedures', 'Functions', 'prestamos', and 'Tables'. The 'Stored Procedures' folder is expanded, showing 'preciosProductos1'. The main editor displays the following SQL code:

```

1 DELIMITER //
2 CREATE PROCEDURE preciosProductos1 (IN tipoProducto VARCHAR(50))
3
4 BEGIN
5     DECLARE Maximo INT;
6     DECLARE Minimo INT;
7     DECLARE Media INT;
8
9     SET Maximo = (SELECT max(valor) from inventario WHERE idProducto = tipoproducto);
10    SET Minimo = (SELECT min(valor) from inventario WHERE idProducto = tipoproducto);
11    SET Media = (SELECT avg(valor) from inventario WHERE idProducto = tipoproducto);
12
13    SELECT Maximo;
14    SELECT Minimo;
15    SELECT Media;
16
17 END;
18 //
19 CALL preciosProductos1 (4);
20

```

The screenshot shows the SQL Studio interface after executing the stored procedure. The main editor displays the following SQL code:

```

1 call myfood.preciosProductos1('4');
2

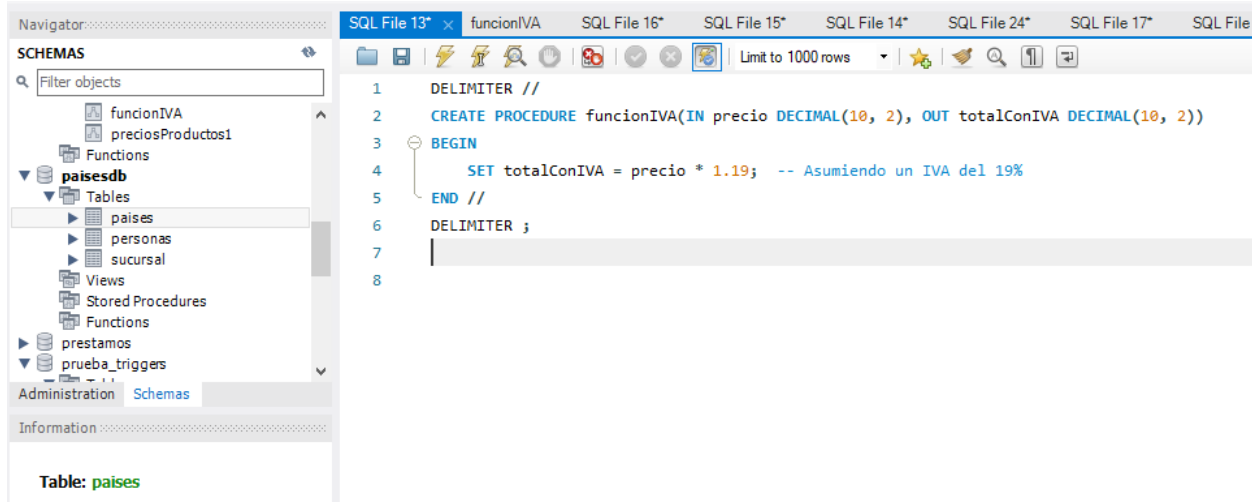
```

The 'Result Grid' pane shows the output of the procedure. It contains a table with one row and one column:

Media
600

Below the table, there is a button labeled 'Resets all sorted columns'. The 'Information' pane at the bottom shows 'Result 1', 'Result 2', and 'Result 3'.

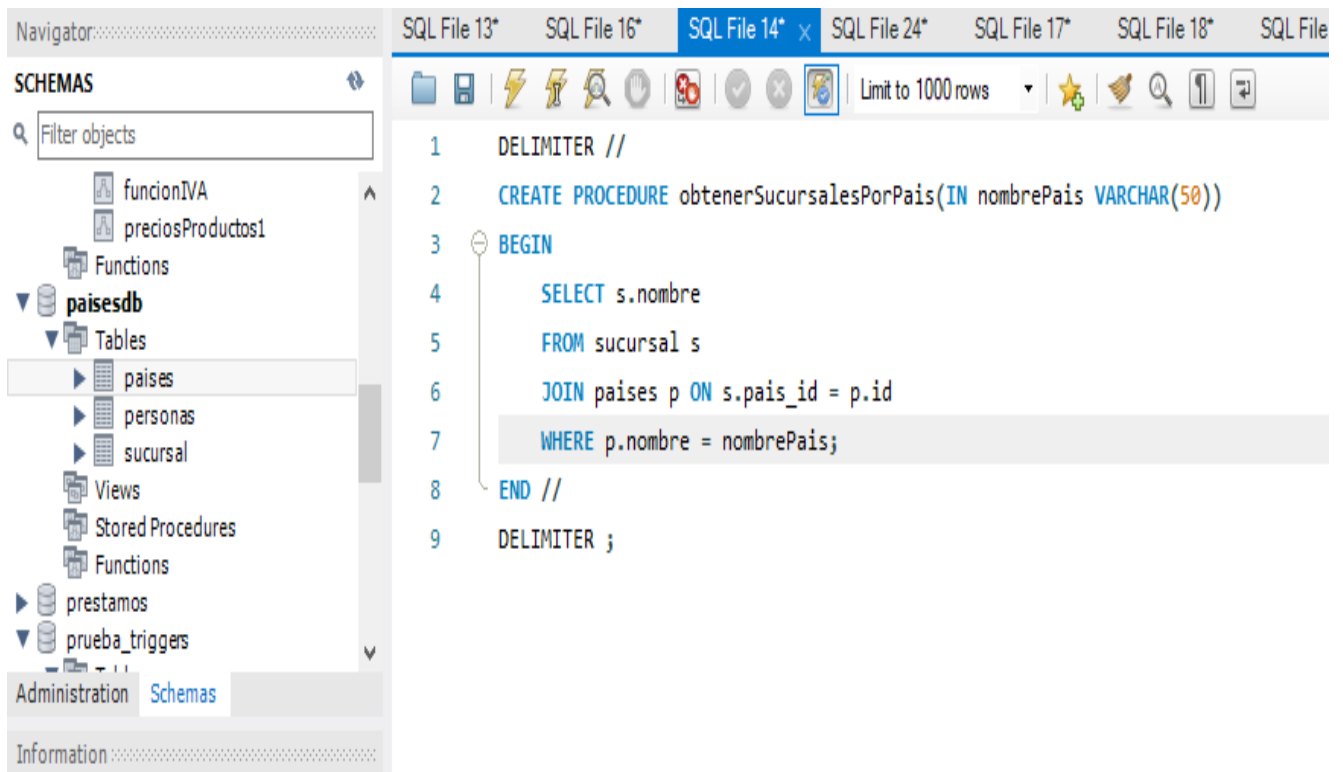
Ejercicio 5. Realice un procedimiento que se llame funcionIVA que incluya una función que calcule el total con el incremento del iva.



The screenshot shows a SQL IDE interface. On the left, the 'SCHEMAS' pane displays a tree view of the database 'paísesdb', including tables like 'países', 'personas', and 'sucursal'. The main editor window, titled 'funcionIVA', contains the following SQL code:

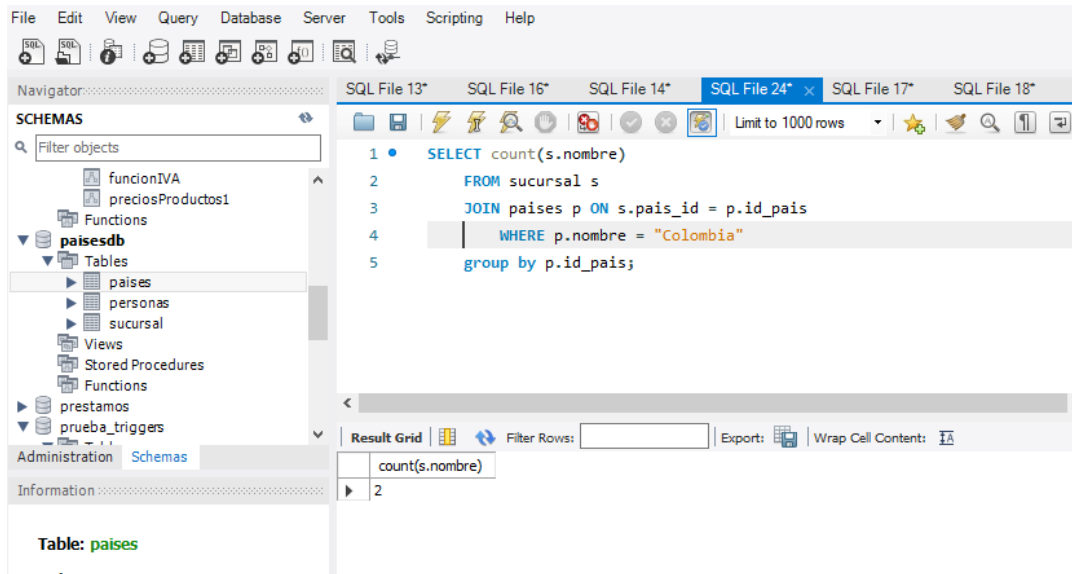
```
1 DELIMITER //
2 CREATE PROCEDURE funcionIVA(IN precio DECIMAL(10, 2), OUT totalConIVA DECIMAL(10, 2))
3 BEGIN
4     SET totalConIVA = precio * 1.19; -- Asumiendo un IVA del 19%
5 END //
6 DELIMITER ;
7
8
```

Ejercicio 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 todas las sucursales que existen en la tabla de ese país.

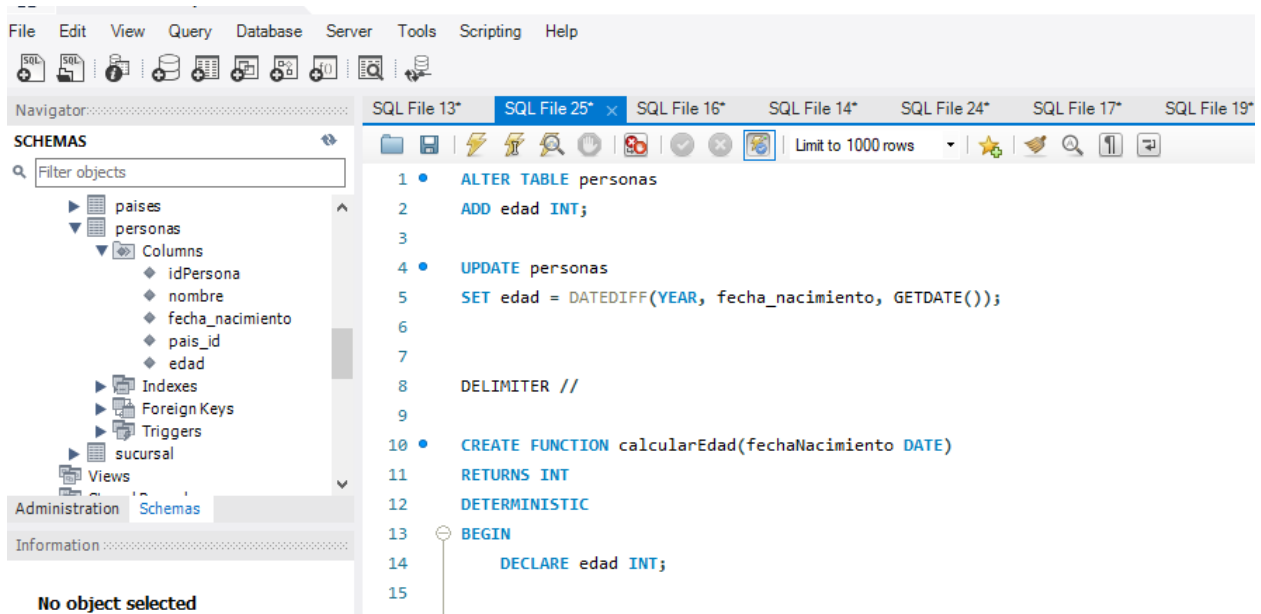


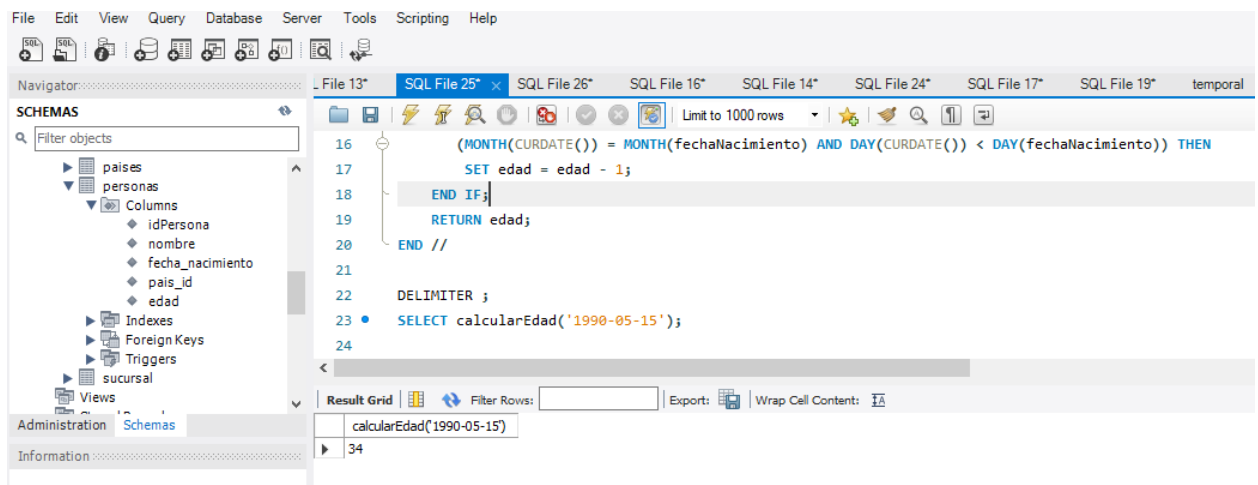
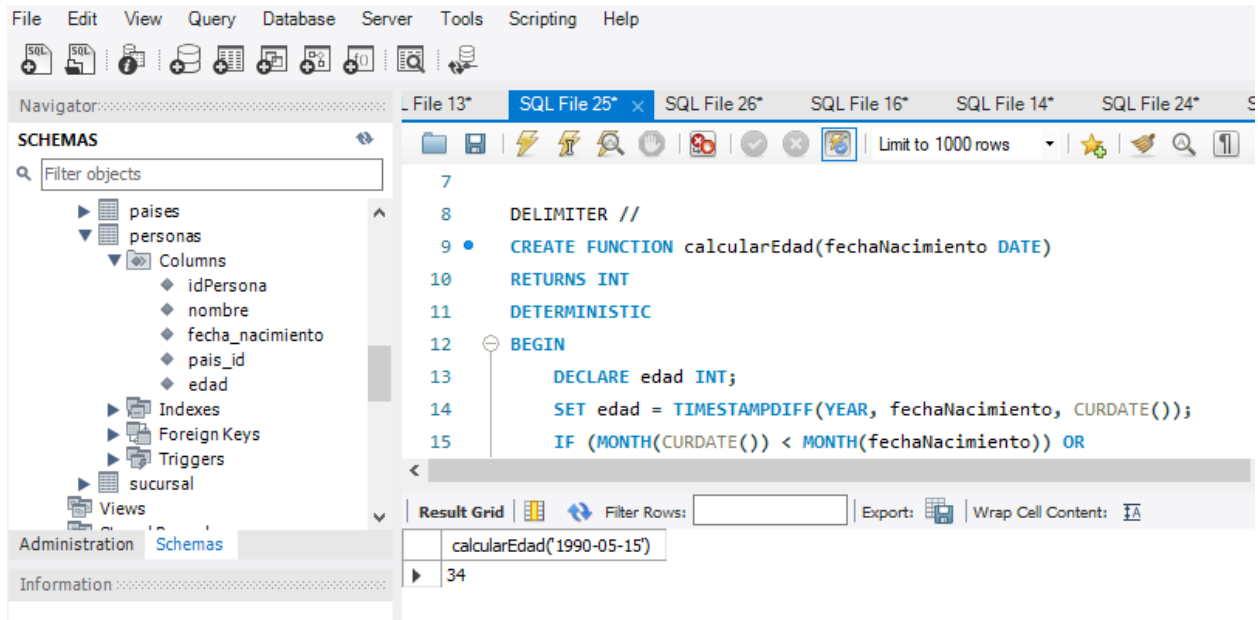
The screenshot shows the same SQL IDE interface. The main editor window, titled 'obtenerSucursalesPorPais', contains the following SQL code:

```
1 DELIMITER //
2 CREATE PROCEDURE obtenerSucursalesPorPais(IN nombrePais VARCHAR(50))
3 BEGIN
4     SELECT s.nombre
5     FROM sucursal s
6     JOIN países p ON s.pais_id = p.id
7     WHERE p.nombre = nombrePais;
8 END //
9 DELIMITER ;
```



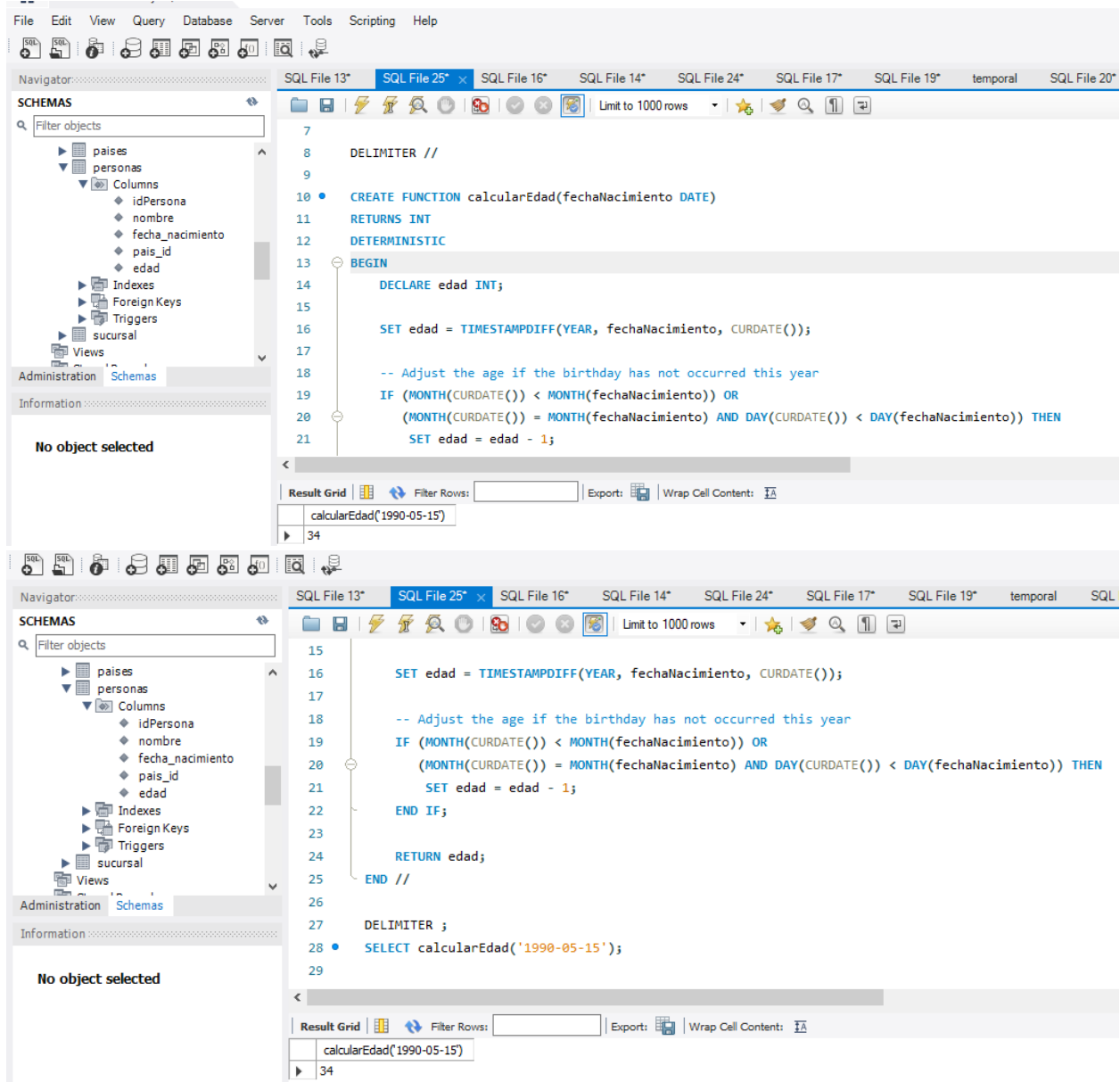
Ejercicio 7. Una vez creada la tabla se decide añadir una nueva columna a la tabla llamada edad que será un valor calculado a partir de la columna fecha_nacimiento. Escriba la sentencia SQL necesaria para modificar la tabla y añadir la nueva columna.





Ejercicio 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.



Ejercicio 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.

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including 'países', 'personas', and 'sucursal'. The 'personas' table is selected, showing its columns: 'idPersona', 'nombre', 'fecha_nacimiento', 'pais_id', and 'edad'. The main editor window shows the SQL script for creating a function:

```
1 DELIMITER //
2
3 CREATE FUNCTION calcularEdad(fechaNacimiento DATE)
4 RETURNS INT
5 DETERMINISTIC
6 BEGIN
7     DECLARE edad INT;
8     SET edad = TIMESTAMPDIFF(YEAR, fechaNacimiento, CURDATE());
9
```

Below the script, the 'Result Grid' shows the output of the function for five rows of data:

nombre	fecha_nacimiento	edad
Mary Rubiano	1994-09-15	30
Juan Rodriguez	1982-04-25	42
Daniel Silva	1986-05-28	38
Diego Gonzalez	2000-03-02	24
Andres Ramirez	1979-10-26	43

The screenshot shows the MySQL Workbench interface with the same database structure as the previous image. The main editor window shows the completion of the function script:

```
10 IF (MONTH(CURDATE()) < MONTH(fechaNacimiento)) OR
11 (MONTH(CURDATE()) = MONTH(fechaNacimiento) AND DAY(CURDATE()) < DAY(fechaNacimiento)) THEN
12     SET edad = edad - 1;
13 END IF;
14
15 RETURN edad;
16 END //
17
18 DELIMITER ;
```

Below the script, the 'Result Grid' shows the output of the function for the same five rows of data as the previous image:

nombre	fecha_nacimiento	edad
Mary Rubiano	1994-09-15	30
Juan Rodriguez	1982-04-25	42
Daniel Silva	1986-05-28	38
Diego Gonzalez	2000-03-02	24
Andres Ramirez	1979-10-26	43

The screenshot shows the SQL Studio interface. On the left, the 'SCHEMAS' pane displays a tree view with 'personas' expanded, showing columns like 'idPersona', 'nombre', 'fecha_nacimiento', 'pais_id', and 'edad'. The main editor shows the following SQL code:

```

14
15     RETURN edad;
16 END //
17
18 DELIMITER ;
19
20 • SELECT nombre, fecha_nacimiento, calcularEdad(fecha_nacimiento) AS edad
21 FROM personas;
22

```

Below the code, the 'Result Grid' shows the output of the query:

nombre	fecha_nacimiento	edad
Mary Rubiano	1994-09-15	30
Juan Rodriguez	1982-04-25	42
Daniel Silva	1986-05-28	38
Diego Gonzalez	2000-03-02	24
Andres Ramirez	1979-10-26	43

Ejercicio10. Escribe un procedimiento almacenado para su proyecto integrador que sea útil.

Bono a profesional con mas clientes atendidos.

The screenshot shows the SQL Studio interface. On the left, the 'SCHEMAS' pane displays a tree view with 'modava' expanded, showing stored procedures like 'cantidadProductos', 'mostrarHolaMundo', 'mostrarNota', 'otorgarBonoProfesionalVariable', and 'otorgarDescuentoClientesFieles'. The main editor shows the following SQL code:

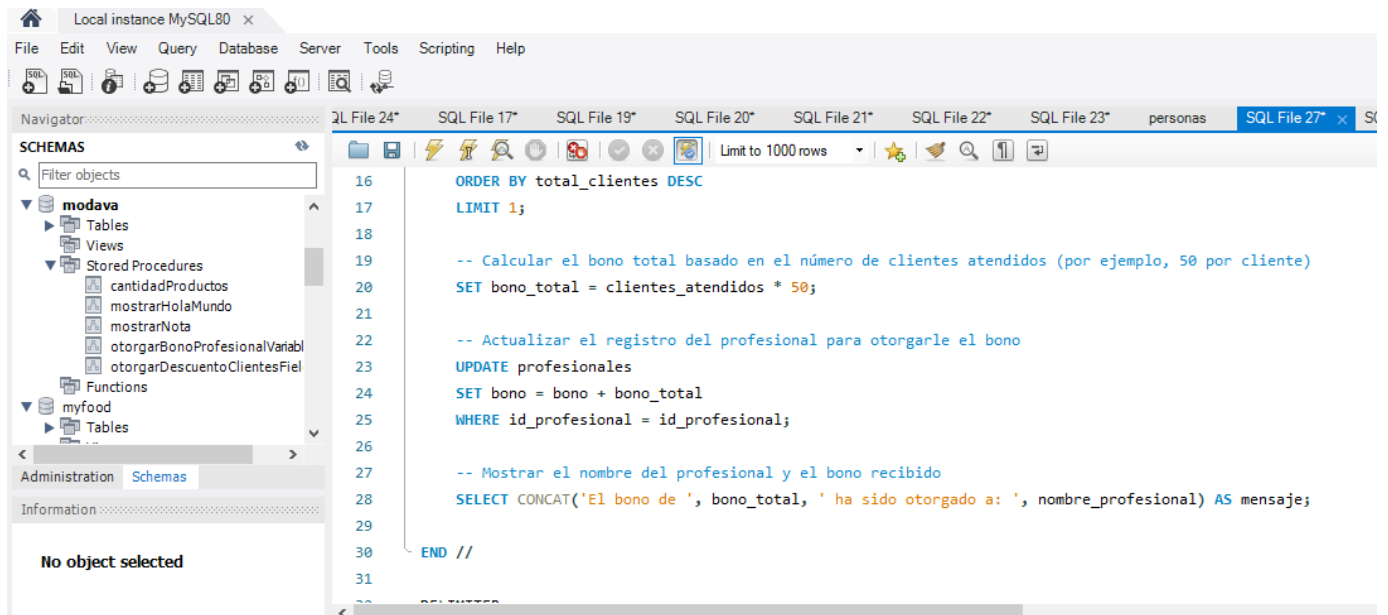
```

1 DELIMITER //
2
3 • CREATE PROCEDURE otorgarBonoProfesionalVariable()
4 BEGIN
5     DECLARE id_profesional INT;
6     DECLARE nombre_profesional VARCHAR(100);
7     DECLARE clientes_atendidos INT;
8     DECLARE bono_total DECIMAL(10, 2);
9
10    -- Seleccionar el profesional que ha atendido más clientes y el número de clientes atendidos
11    SELECT p.id_profesional, p.nombre, COUNT(s.id_cliente) AS total_clientes
12    INTO id_profesional, nombre_profesional, clientes_atendidos
13    FROM profesionales p
14    JOIN servicios s ON p.id_profesional = s.id_profesional
15    GROUP BY p.id_profesional
16    ORDER BY total_clientes DESC
17

```

Below the code, the 'Output' pane shows the execution results:

#	Time	Action	Message
1	21:29:16	DROP PROCEDURE 'modava'. 'otorgarBonoProfesional'	0 row(s) affected
2	21:29:22	DROP PROCEDURE 'modava'. 'otorgarDescuentoClientesFieles'	0 row(s) affected
3	21:29:34	CREATE PROCEDURE otorgarBonoProfesionalVariable() BEGIN DECLARE id_profesio...	0 row(s) affected
4	21:30:29	CREATE PROCEDURE otorgarDescuentoClientesFielesVariable() BEGIN DECLARE id_...	0 row(s) affected



Programa descuento cliente leales.

The first screenshot shows the MySQL Workbench interface with a SQL script being written in a file named 'SQL File 24*'. The script defines a stored procedure named 'otorgarDescuentoClientesFielesVariable()' which takes no arguments. It declares variables for client ID, name, total visits, and discount. It then selects the top 5 most loyal clients (those with the most visits) and calculates a discount for them based on their number of visits (2% per visit, up to 20%).

```

1  DELIMITER //
2
3  CREATE PROCEDURE otorgarDescuentoClientesFielesVariable()
4  BEGIN
5      DECLARE id_cliente INT;
6      DECLARE nombre_cliente VARCHAR(100);
7      DECLARE total_visitas INT;
8      DECLARE descuento DECIMAL(5, 2);
9
10     -- Seleccionar los clientes más fieles y el número de visitas que han hecho
11     SELECT c.id_cliente, c.nombre, COUNT(v.id_visita) AS total_visitas
12     INTO id_cliente, nombre_cliente, total_visitas
13     FROM clientes c
14     JOIN visitas v ON c.id_cliente = v.id_cliente
15     GROUP BY c.id_cliente
16     HAVING total_visitas >= 5; -- Clientes con al menos 5 visitas

```

The second screenshot shows the same MySQL Workbench interface, but now the script is being executed. The 'Output' tab at the bottom shows the results of the execution, including the creation of the stored procedure and the calculation of the discount for the selected clients.

#	Time	Action	Message
1	21:29:16	DROP PROCEDURE 'modava`.`otorgarBonoProfesional'	0 row(s) affected
2	21:29:22	DROP PROCEDURE 'modava`.`otorgarDescuentoClientesFieles'	0 row(s) affected
3	21:29:34	CREATE PROCEDURE otorgarBonoProfesionalVariable() BEGIN DECLARE id_profesio...	0 row(s) affected
4	21:30:29	CREATE PROCEDURE otorgarDescuentoClientesFielesVariable() BEGIN DECLARE id_...	0 row(s) affected

Conclusión: El uso de **procedimientos almacenados** en bases de datos no solo reduce la complejidad del código en las aplicaciones que interactúan con la base de datos, sino que también mejora el rendimiento y la seguridad al centralizar la lógica de negocio en el servidor de la base de datos. Esto permite un control más preciso sobre las

transacciones y procesos repetitivos, minimizando el riesgo de errores humanos y garantizando la integridad de los datos.

Este trabajo ha contribuido al desarrollo de habilidades clave en el diseño e implementación de soluciones eficientes basadas en bases de datos, resaltando el valor de los procedimientos almacenados como una herramienta fundamental para cualquier proyecto de software moderno. Así, se fortalece la capacidad de los futuros desarrolladores para enfrentar los desafíos de la industria del software, aplicando principios sólidos de programación de bases de datos.

Finalmente, la experiencia adquirida a lo largo de este trabajo refuerza la importancia de la formación técnica y teórica para resolver problemas reales mediante la aplicación de tecnologías de bases de datos, consolidando el aprendizaje en la gestión de datos, automatización de procesos y diseño eficiente de sistemas de información.