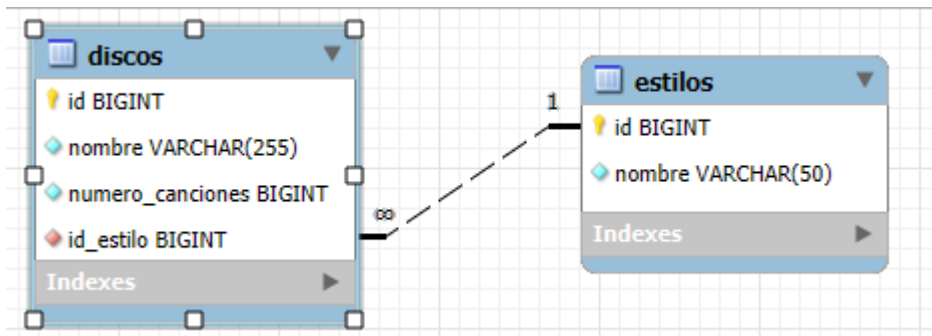


UF2175



Script creación de la BD:

```
-- Crear base de datos\CREATE DATABASE bd_discos;
CREATE DATABASE bd_discos;
USE bd_discos;
-- Crear tabla de estilos
CREATE TABLE estilos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL
);
-- Crear tabla de discos
CREATE TABLE discos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    numero_canciones INT NOT NULL,
    id_estilo INT,
    FOREIGN KEY (id_estilo) REFERENCES estilos(id)
);
-- Crear tabla de log de inserciones en discos
CREATE TABLE log_discos (
    id_log INT AUTO_INCREMENT PRIMARY KEY,
    id_disco INT,
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

-- Insertar datos de ejemplo en estilos

INSERT INTO estilos (nombre) VALUES

('Rock'),

('Pop'),

('Jazz'),

('Metal');

-- Insertar datos de ejemplo en discos

INSERT INTO discos (nombre, numero_canciones, id_estilo) VALUES

('Greatest Hits', 12, 1),

('Love Songs', 9, 2),

('Smooth Jazz', 15, 3),

('Heavy Metal Legends', 8, 4),

('Pop Classics', 11, 2);

UF2176

1. Escribe una consulta para seleccionar todos los discos que tienen más de 10 canciones.

The screenshot shows the MySQL Workbench interface. The 'Query' tab is active, displaying the following SQL code:

```
40 ('Pop Classics', 11, 2);
41
42 -- Consultas UF2176
43 -- 1. Discos con más de 10 canciones
44 SELECT * FROM discos WHERE numero_canciones > 10;
```

The 'Result Grid' shows the results of the query:

#	id	nombre	numero_canciones	id_estilo
1	1	Greatest Hits	12	1
2	3	Smooth Jazz	15	3
5	5	Pop Classics	11	2

The 'Output' tab shows the execution log:

#	Time	Action	Message	Duration / Fetch
7	08:51:00	INSERT INTO estilos (nombre) VALUES ('Rock'), ('Pop'), ('Jazz'), ('Metal')	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.016 sec
8	08:51:00	INSERT INTO discos (nombre, numero_canciones, id_estilo) VALUES ('Greatest Hits', 12, 1), ('Love Songs', 9, 2), ('Smooth Jazz', 15, ...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.016 sec
9	08:51:00	SELECT * FROM discos WHERE numero_canciones > 10 LIMIT 0, 1000	3 row(s) returned	0.000 sec / ...
10	08:51:00	SELECT * FROM discos ORDER BY numero_canciones DESC LIMIT 0, 1000	5 row(s) returned	0.016 sec / ...
11	08:51:01	SELECT id_estilo, COUNT(*) AS cantidad_discos FROM discos GROUP BY id_estilo LIMIT 0, 1000	4 row(s) returned	0.000 sec / ...
12	08:51:01	SELECT AVG(numero_canciones) AS promedio_canciones FROM discos LIMIT 0, 1000	1 row(s) returned	0.000 sec / ...

2. Escribe una consulta para seleccionar todos los discos y ordenarlos por el número de canciones en orden descendente

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL query:

```
-- 2. Discos ordenados por número de canciones descendente
SELECT * FROM discos ORDER BY numero_canciones DESC;
```

The Results Grid displays the following data:

id	nombre	numero_canciones	id_estilo
3	Smooth Jazz	15	3
1	Greatest Hits	12	1
5	Pop Classics	11	2
2	Love Songs	9	2
4	Heavy Metal Legends	8	4

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
7	00:51:00	INSERT INTO estilos (nombre) VALUES ('Rock'), ('Pop'), ('Jazz'), ('Metal')	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0	0.016 sec
8	00:51:00	INSERT INTO discos (nombre, numero_canciones, id_estilo) VALUES ('Greatest Hits', 12, 1), ('Love Songs', 9, 2), ('Smooth Jazz', 15, 3)	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.016 sec
9	00:51:00	SELECT * FROM discos WHERE numero_canciones > 10 LIMIT 0, 1000	3 row(s) returned	0.000 sec
10	00:51:00	SELECT * FROM discos ORDER BY numero_canciones DESC LIMIT 0, 1000	5 row(s) returned	0.016 sec

3. Escribe una consulta para contar el número de discos por cada estilo.

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL query:

```
-- 3. Contar número de discos por estilo
SELECT id_estilo, COUNT(*) AS cantidad_discos
FROM discos
GROUP BY id_estilo;
```

The Results Grid displays the following data:

id_estilo	cantidad_discos
1	1
2	2
3	1
4	1

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
10	00:51:00	SELECT * FROM discos ORDER BY numero_canciones DESC LIMIT 0, 1000	5 row(s) returned	0.016 sec
11	00:51:01	SELECT id_estilo, COUNT(*) AS cantidad_discos FROM discos GROUP BY id_estilo LIMIT 0, 1000	4 row(s) returned	0.000 sec

4. Escribe una consulta para calcular el promedio de canciones por disco.

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
52 GROUP BY id_estilo;
53
54 -- 4. Calcular promedio de canciones
55 * SELECT AVG(numero_canciones) AS promedio_canciones
56 FROM discos;
```

The 'Result Grid' shows the result of the query:

promedio_canciones
11.0000

The 'Output' pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
10	08:51:00	SELECT * FROM discos ORDER BY numero_canciones DESC LIMIT 0, 1000	5 row(s) returned	0.016 sec / ...
11	08:51:01	SELECT id_estilo, COUNT(*) AS cantidad_discos FROM discos GROUP BY id_estilo LIMIT 0, 1000	4 row(s) returned	0.000 sec / ...
12	08:51:01	SELECT AVG(numero_canciones) AS promedio_canciones FROM discos LIMIT 0, 1000	1 row(s) returned	0.000 sec / ...

5. Escribe una consulta para encontrar el disco con el mayor número de canciones.

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
58 -- 5. Disco con mayor número de canciones
59 * SELECT * FROM discos
60 ORDER BY numero_canciones DESC
61 LIMIT 1;
```

The 'Result Grid' shows the result of the query:

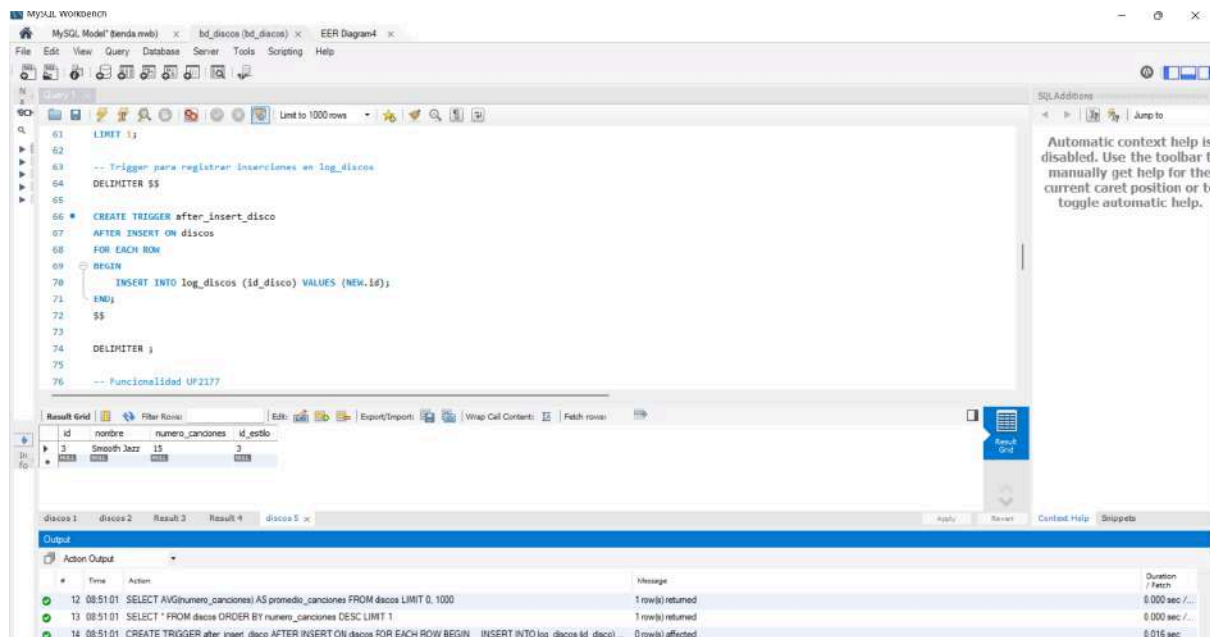
id	nombre	numero_canciones	id_estilo
3	Smooth Jazz	15	3

The 'Output' pane shows the execution log:

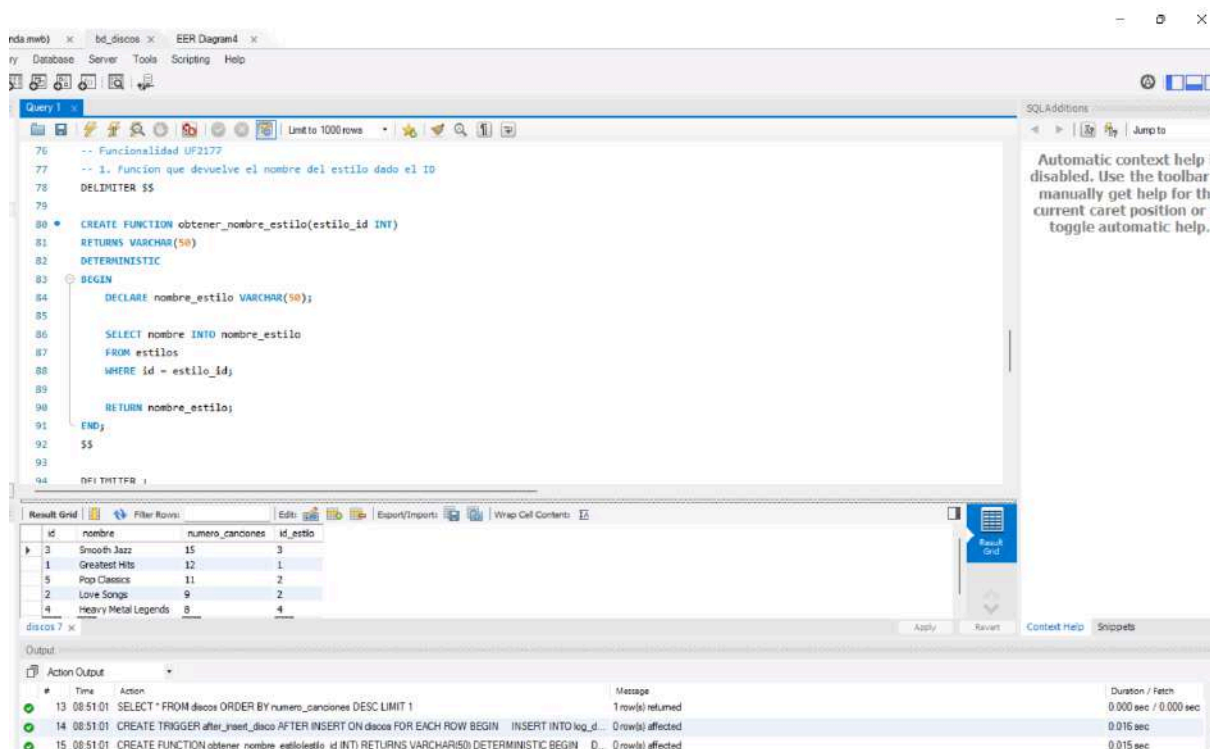
#	Time	Action	Message	Duration / Fetch
10	08:51:00	SELECT * FROM discos ORDER BY numero_canciones DESC LIMIT 0, 1000	5 row(s) returned	0.016 sec / ...
11	08:51:01	SELECT id_estilo, COUNT(*) AS cantidad_discos FROM discos GROUP BY id_estilo LIMIT 0, 1000	4 row(s) returned	0.000 sec / ...
12	08:51:01	SELECT AVG(numero_canciones) AS promedio_canciones FROM discos LIMIT 0, 1000	1 row(s) returned	0.000 sec / ...
13	08:51:01	SELECT * FROM discos ORDER BY numero_canciones DESC LIMIT 1	1 row(s) returned	0.000 sec / ...

UF2177

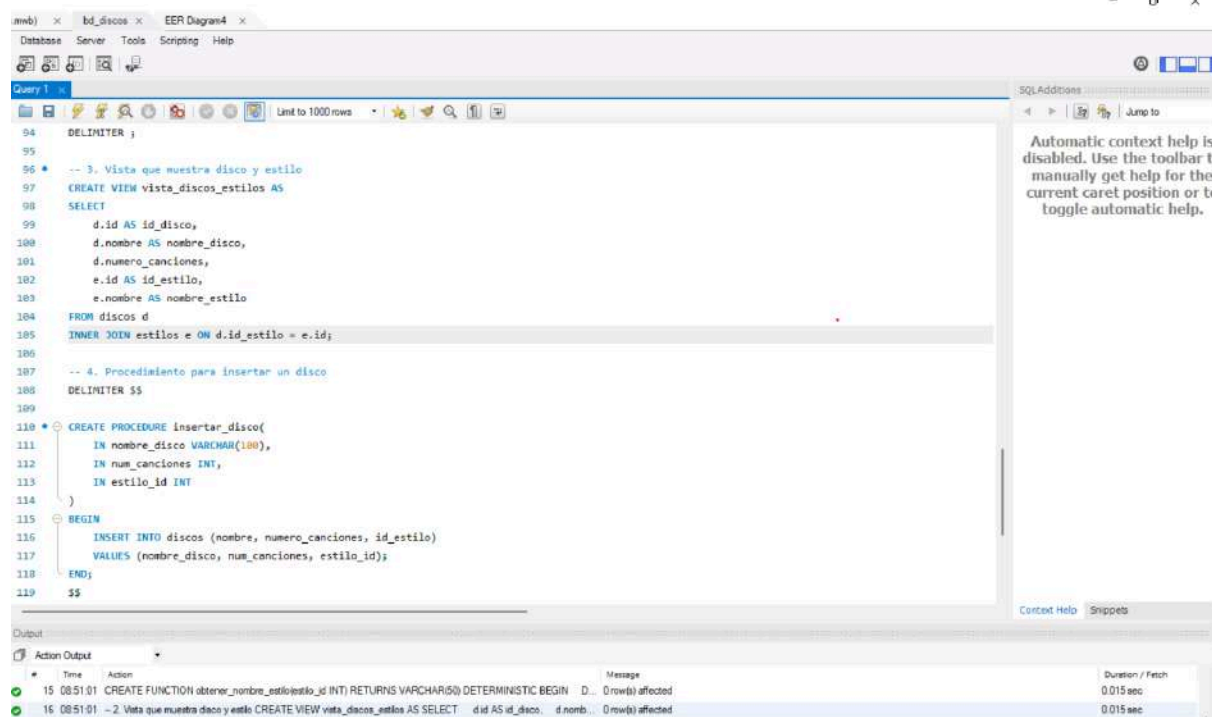
1. Escribe un trigger que registre en una tabla log_discos cada vez que se inserte un nuevo disco en la tabla discos.



2. Escribe una función que devuelva el nombre del estilo dado ID estilo.



3. Escribe una vista que muestre el ID del disco, el nombre, el numero de canciones, el ID del estilo y el nombre del estilo.



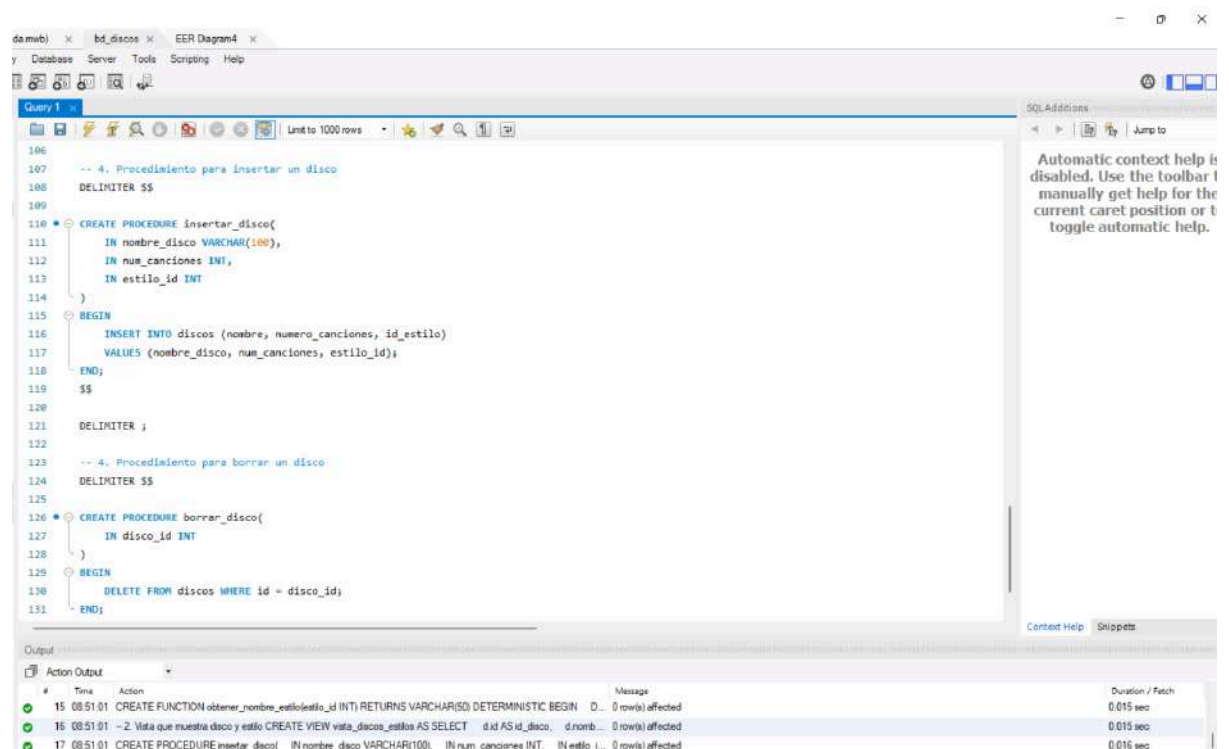
The screenshot shows the SQL Server Enterprise Manager interface. The main window displays a SQL script for creating a view. The script is as follows:

```
DELIMITER ;
-- 3. Vista que muestra disco y estilo
CREATE VIEW vista_discos_estilos AS
SELECT
    d.id AS id_disco,
    d.nombre AS nombre_disco,
    d.numero_canciones,
    e.id AS id_estilo,
    e.nombre AS nombre_estilo
FROM discos d
INNER JOIN estilos e ON d.id_estilo = e.id;
DELIMITER $$
```

The Output window at the bottom shows the execution results:

#	Time	Action	Message	Duration / Fetch
15	08:51:01	CREATE FUNCTION obtener_nombre_estilo(estilo_id INT) RETURNS VARCHAR(50) DETERMINISTIC BEGIN	D... 0 row(s) affected	0.015 sec
16	08:51:01	-- 2. Vista que muestra disco y estilo CREATE VIEW vista_discos_estilos AS SELECT	d.id AS id_disco, d.nombre... 0 row(s) affected	0.015 sec

4. ESCRIBE UN PROCEDIMIENTO ALMACENADO QUE INSERTA UN DISCO PASANDOLE EL NOMBRE, EL NÚMERO DE CANCIONES EL ID DEL ESTILO Y EL NOMBRE



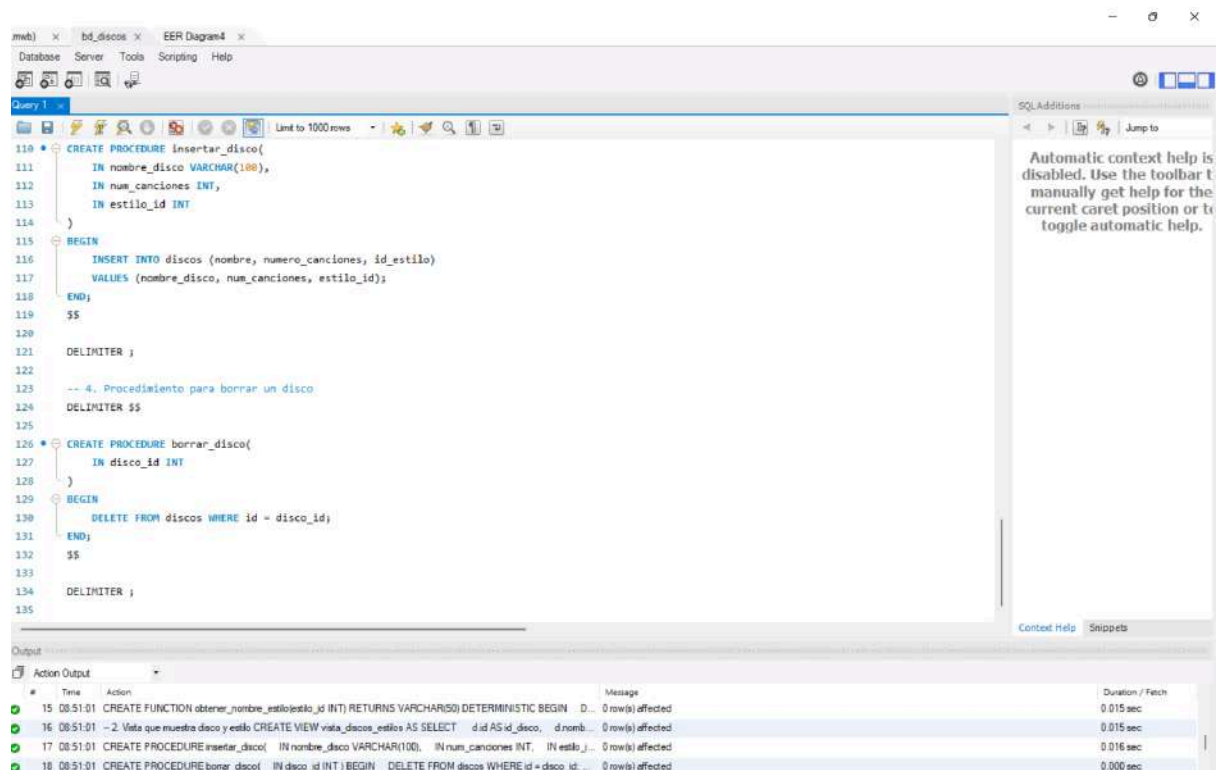
The screenshot shows the SQL Server Enterprise Manager interface. The main window displays a SQL script for creating a stored procedure. The script is as follows:

```
DELIMITER $$
CREATE PROCEDURE insertar_disco(
    IN nombre_disco VARCHAR(100),
    IN num_canciones INT,
    IN estilo_id INT
)
BEGIN
    INSERT INTO discos (nombre, numero_canciones, id_estilo)
    VALUES (nombre_disco, num_canciones, estilo_id);
END;
DELIMITER ;
-- 4. Procedimiento para borrar un disco
DELIMITER $$
CREATE PROCEDURE borrar_disco(
    IN disco_id INT
)
BEGIN
    DELETE FROM discos WHERE id = disco_id;
END;
DELIMITER ;
```

The Output window at the bottom shows the execution results:

#	Time	Action	Message	Duration / Fetch
15	08:51:01	CREATE FUNCTION obtener_nombre_estilo(estilo_id INT) RETURNS VARCHAR(50) DETERMINISTIC BEGIN	D... 0 row(s) affected	0.015 sec
16	08:51:01	-- 2. Vista que muestra disco y estilo CREATE VIEW vista_discos_estilos AS SELECT	d.id AS id_disco, d.nombre... 0 row(s) affected	0.015 sec
17	08:51:01	CREATE PROCEDURE insertar_disco(IN nombre_disco VARCHAR(100), IN num_canciones INT, IN estilo_id INT)	0 row(s) affected	0.016 sec

5. ESCRIBE UN PROCEDIMIENTO ALMACENADO QUE BORRE UN DISCO PASANDOLE EL ID



```
110 CREATE PROCEDURE Insertar_disco(  
111     IN nombre_disco VARCHAR(100),  
112     IN num_canciones INT,  
113     IN estilo_id INT  
114 )  
115 BEGIN  
116     INSERT INTO discos (nombre, numero_canciones, id_estilo)  
117     VALUES (nombre_disco, num_canciones, estilo_id);  
118 END;  
119 $$  
120  
121 DELIMITER ;  
122  
123 -- 4. Procedimiento para borrar un disco.  
124 DELIMITER $$  
125  
126 CREATE PROCEDURE borrar_disco(  
127     IN disco_id INT  
128 )  
129 BEGIN  
130     DELETE FROM discos WHERE id = disco_id;  
131 END;  
132 $$  
133  
134 DELIMITER ;  
135
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

#	Time	Action	Message	Duration / Fetch
15	08:51:01	CREATE FUNCTION obtener_nombre_estilo(estilo_id INT) RETURNS VARCHAR(50) DETERMINISTIC BEGIN	0 row(s) affected	0.015 sec
16	08:51:01	-- Vista que muestra disco y estilo CREATE VIEW vista_discos_estilo AS SELECT d.id AS id_disco, d.nombre	0 row(s) affected	0.015 sec
17	08:51:01	CREATE PROCEDURE Insertar_disco(IN nombre_disco VARCHAR(100), IN num_canciones INT, IN estilo_id	0 row(s) affected	0.016 sec
18	08:51:01	CREATE PROCEDURE borrar_disco(IN disco_id INT) BEGIN DELETE FROM discos WHERE id = disco_id;	0 row(s) affected	0.000 sec

-- Trigger para registrar inserciones en log_discos
DELIMITER \$\$

```
CREATE TRIGGER after_insert_disco  
AFTER INSERT ON discos  
FOR EACH ROW  
BEGIN  
    INSERT INTO log_discos (id_disco) VALUES (NEW.id);  
END;  
$$  
  
DELIMITER ;
```

```
-- Funcionalidad UF2177
-- 1. Función que devuelve el nombre del estilo dado el ID
DELIMITER $$
```

```
CREATE FUNCTION obtener_nombre_estilo(estilo_id INT)
RETURNS VARCHAR(50)
DETERMINISTIC
BEGIN
    DECLARE nombre_estilo VARCHAR(50);

    SELECT nombre INTO nombre_estilo
    FROM estilos
    WHERE id = estilo_id;

    RETURN nombre_estilo;
END;
$$
```

```
DELIMITER ;
```

```
-- 2. Vista que muestra disco y estilo
CREATE VIEW vista_discos_estilos AS
SELECT
    d.id AS id_disco,
    d.nombre AS nombre_disco,
    d.numero_canciones,
    e.id AS id_estilo,
    e.nombre AS nombre_estilo
FROM discos d
INNER JOIN estilos e ON d.id_estilo = e.id;
```


-- 3. Procedimiento para insertar un disco

DELIMITER \$\$

CREATE PROCEDURE insertar_disco(

IN nombre_disco VARCHAR(100),

IN num_canciones INT,

IN estilo_id INT

)

BEGIN

INSERT INTO discos (nombre, numero_canciones, id_estilo)

VALUES (nombre_disco, num_canciones, estilo_id);

END;

\$\$

DELIMITER ;

-- 4. Procedimiento para borrar un disco

DELIMITER \$\$

CREATE PROCEDURE borrar_disco(

IN disco_id INT

)

BEGIN

DELETE FROM discos WHERE id = disco_id;

END;

\$\$

DELIMITER ;

DIAGRAMA FINAL:

