

## Actividad 28

Crea un procedimiento que reciba el `customer_id` y devuelva el nombre completo (`first_name + last_name`) como salida.

The screenshot shows the SQL Developer interface. The main window displays the following SQL code:

```
3 DROP PROCEDURE IF EXISTS obtener_nombre_completo
4 DELIMITER $
5 CREATE PROCEDURE obtener_nombre_completo(IN id_cliente INT, OUT nombre_completo VARCHAR(100))
6 BEGIN
7     SELECT CONCAT(first_name, ' ', last_name)
8     INTO nombre_completo
9     FROM customer
10    WHERE customer_id = id_cliente;
11 END $
12 DELIMITER ;
13
14 CALL obtener_nombre_completo(1, @nombre);
```

The **Result Grid** shows the output of the procedure call:

	@nombre
1	MARY SMITH

The **Output** pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
10	07:30:28	DROP PROCEDURE IF EXISTS obtener_nombre_completo	0 row(s) affected	0.032 sec
11	07:30:34	USE sakila	0 row(s) affected	0.000 sec
12	07:30:34	CREATE PROCEDURE obtener_nombre_completo(IN id_cliente INT, OUT nombre_com...	0 row(s) affected	0.031 sec
13	07:30:34	CALL obtener_nombre_completo(1, @nombre)	1 row(s) affected	0.078 sec
14	07:30:34	SELECT @nombre LIMIT 0, 500	1 row(s) returned	0.000 sec / 0.000 sec

Crear un procedimiento que reciba el nombre de una categoría (por ejemplo, 'Action') y devuelva cuántas películas hay en esa categoría.

The screenshot shows the SQL Developer interface. The main window displays the following SQL code:

```
17
18 DROP PROCEDURE IF EXISTS contar_peliculas_por_categoria
19 DELIMITER $
20 CREATE PROCEDURE contar_peliculas_por_categoria(IN nombre_categoria VARCHAR(50), OUT total INT)
21 BEGIN
22     SELECT COUNT(*)
23     INTO total
24     FROM film_category fc
25     JOIN category c ON fc.category_id = c.category_id
26     WHERE c.name = nombre_categoria;
27 END $
28 DELIMITER ;
```

The **Result Grid** shows the output of the procedure call:

	@total
1	64

The **Output** pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
13	07:30:34	CALL obtener_nombre_completo(1, @nombre)	1 row(s) affected	0.078 sec
14	07:30:34	SELECT @nombre LIMIT 0, 500	1 row(s) returned	0.000 sec / 0.000 sec
15	07:32:03	CREATE PROCEDURE contar_peliculas_por_categoria(IN nombre_categoria VARCHA...	0 row(s) affected	0.016 sec
16	07:32:03	CALL contar_peliculas_por_categoria('Action', @total)	1 row(s) affected	0.000 sec
17	07:32:03	SELECT @total LIMIT 0, 500	1 row(s) returned	0.000 sec / 0.000 sec

Crea un procedimiento que reciba un customer\_id y devuelva 1 si existe, o 0 si no.

The screenshot shows a SQL IDE interface with a query editor, a result grid, and an action output pane. The query editor contains the following SQL code:

```
35 DELIMITER $
36 CREATE PROCEDURE existe_cliente(IN id_cliente INT, OUT existe TINYINT)
37 BEGIN
38     SELECT COUNT(*) > 0
39     INTO existe
40     FROM customer
41     WHERE customer_id = id_cliente;
42 END $
43 DELIMITER ;
44 CALL existe_cliente(1, @existe);
45 SELECT @existe;
46
```

The result grid shows the output of the query, which is a single row with the value 1.

The action output pane shows the execution log, including the creation of the procedure and the call to the procedure.

#	Time	Action	Message	Duration / Fetch
16	07:32:03	CALL contar_peliculas_por_categoria('Action', @total)	1 row(s) affected	0.000 sec
17	07:32:03	SELECT @total LIMIT 0, 500	1 row(s) returned	0.000 sec / 0.000 sec
18	07:32:59	CREATE PROCEDURE existe_cliente(IN id_cliente INT, OUT existe TINYINT) BEGIN ...	0 row(s) affected	0.000 sec
19	07:32:59	CALL existe_cliente(1, @existe)	1 row(s) affected	0.000 sec
20	07:32:59	SELECT @existe LIMIT 0, 500	1 row(s) returned	0.015 sec / 0.000 sec

Crear un procedimiento que reciba un número (min\_duracion) y muestre todas las películas que duran más que ese valor.

The screenshot shows a SQL IDE interface with a query editor, a result grid, and an action output pane. The query editor contains the following SQL code:

```
49
50 DELIMITER $
51 CREATE PROCEDURE peliculas_mayor_duracion(IN min_duracion INT)
52 BEGIN
53     SELECT title, length
54     FROM film
55     WHERE length > min_duracion;
56 END $
57 DELIMITER ;
58 CALL peliculas_mayor_duracion(120);
59
60
```

The result grid shows the output of the query, which is a list of movie titles and their lengths.

title	length
AFRICAN EGG	130
AGENT TRUMAN	169
ALAMO VIDEOTAPE	126
ALASKA PHANTOM	136
ALI FOREVER	150
ALLEY EVOLUTION	180

The action output pane shows the execution log, including the creation of the procedure and the call to the procedure.

#	Time	Action	Message	Duration / Fetch
18	07:32:59	CREATE PROCEDURE existe_cliente(IN id_cliente INT, OUT existe TINYINT) BEGIN ...	0 row(s) affected	0.000 sec
19	07:32:59	CALL existe_cliente(1, @existe)	1 row(s) affected	0.000 sec
20	07:32:59	SELECT @existe LIMIT 0, 500	1 row(s) returned	0.015 sec / 0.000 sec
21	07:33:24	CREATE PROCEDURE peliculas_mayor_duracion(IN min_duracion INT) BEGIN SELE...	0 row(s) affected	0.015 sec
22	07:33:24	CALL peliculas_mayor_duracion(120)	457 row(s) returned	0.000 sec / 0.000 sec

Crea un procedimiento que reciba el customer\_id y devuelva una lista de títulos de películas que ha rentado.

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

```

DELIMITER $
CREATE PROCEDURE peliculas_rentadas_por_cliente(IN id_cliente INT)
BEGIN
    SELECT DISTINCT f.title
    FROM rental r
    JOIN inventory i ON r.inventory_id = i.inventory_id
    JOIN film f ON i.film_id = f.film_id
    WHERE r.customer_id = id_cliente;
END $
DELIMITER ;
CALL peliculas_rentadas_por_cliente(1);

```

The Results pane shows the output of the procedure call, displaying a list of movie titles:

title
PATIENT SISTER
TALENTED HOMICIDE
MUSKETEERS WAIT
DETECTIVE VISION
FERRIS MOTHER
CLOSER BANG

The Output pane shows the execution log, including the creation of the procedure and the results of the call:

#	Time	Action	Message	Duration / Fetch
20	07:32:59	SELECT @existe LIMIT 0, 500	1 row(s) returned	0.015 sec / 0.000 sec
21	07:33:24	CREATE PROCEDURE peliculas_mayor_duracion(IN min_duracion INT) BEGIN SELE...	0 row(s) affected	0.015 sec
22	07:33:24	CALL peliculas_mayor_duracion(120)	457 row(s) returned	0.000 sec / 0.000 sec
23	07:33:57	CREATE PROCEDURE peliculas_rentadas_por_cliente(IN id_cliente INT) BEGIN SEL...	0 row(s) affected	0.032 sec
24	07:33:57	CALL peliculas_rentadas_por_cliente(1)	30 row(s) returned	0.015 sec / 0.000 sec

Crear un procedimiento que reciba el store\_id y devuelva el total de dinero generado por esa tienda.

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

```

DELIMITER $
CREATE PROCEDURE total_dinero_por_tienda(IN id_tienda INT, OUT total DECIMAL(10,2))
BEGIN
    SELECT SUM(p.amount)
    INTO total
    FROM payment p
    JOIN rental r ON p.rental_id = r.rental_id
    JOIN inventory i ON r.inventory_id = i.inventory_id
    WHERE i.store_id = id_tienda;
END $
DELIMITER ;
CALL total_dinero_por_tienda(1, @total);
SELECT @total LIMIT 0, 500;

```

The Results pane shows the output of the procedure call, displaying the total amount generated by the store:

@total
33679.79

The Output pane shows the execution log, including the creation of the procedure and the results of the call:

#	Time	Action	Message	Duration / Fetch
23	07:33:57	CREATE PROCEDURE peliculas_rentadas_por_cliente(IN id_cliente INT) BEGIN SEL...	0 row(s) affected	0.032 sec
24	07:33:57	CALL peliculas_rentadas_por_cliente(1)	30 row(s) returned	0.015 sec / 0.000 sec
25	07:34:37	CREATE PROCEDURE total_dinero_por_tienda(IN id_tienda INT, OUT total DECIMAL(1...	0 row(s) affected	0.016 sec
26	07:34:37	CALL total_dinero_por_tienda(1, @total)	1 row(s) affected	0.094 sec
27	07:34:38	SELECT @total LIMIT 0, 500	1 row(s) returned	0.000 sec / 0.000 sec

Crear un procedimiento que reciba el `film_id` y devuelva el número de copias disponibles en la tabla `inventory`.

The screenshot shows a SQL IDE interface with a query editor, a result grid, and an action output pane.

**Query 1 SQL File 5\***

```

91
92 DELIMITER $
93 CREATE PROCEDURE copias_por_pelicula(IN id_pelicula INT, OUT cantidad INT)
94 BEGIN
95     SELECT COUNT(*)
96     INTO cantidad
97     FROM inventory
98     WHERE film_id = id_pelicula;
99 END $
100 DELIMITER ;
101 CALL copias_por_pelicula(10, @cantidad);
102 SELECT @cantidad;

```

**Result Grid**

cantidad
7

**Action Output**

#	Time	Action	Message	Duration / Fetch
26	07:34:37	CALL total_dinero_por_tienda(1, @total)	1 row(s) affected	0.094 sec
27	07:34:38	SELECT @total LIMIT 0, 500	1 row(s) returned	0.000 sec / 0.000 sec
28	07:36:09	CREATE PROCEDURE copias_por_pelicula(IN id_pelicula INT, OUT cantidad INT) BEG...	0 row(s) affected	0.031 sec
29	07:36:09	CALL copias_por_pelicula(10, @cantidad)	1 row(s) affected	0.000 sec
30	07:36:09	SELECT @cantidad LIMIT 0, 500	1 row(s) returned	0.000 sec / 0.000 sec

Crear un procedimiento que reciba el `film_id` y muestre una lista con los nombres de los actores que participan en esa película.

The screenshot shows a SQL IDE interface with a query editor, a result grid, and an action output pane.

**Query 1 SQL File 5\***

```

106
107 DELIMITER $
108 CREATE PROCEDURE actores_por_pelicula(IN id_pelicula INT)
109 BEGIN
110     SELECT CONCAT(a.first_name, ' ', a.last_name) AS nombre_actor
111     FROM film_actor fa
112     JOIN actor a ON fa.actor_id = a.actor_id
113     WHERE fa.film_id = id_pelicula;
114 END $
115 DELIMITER ;
116 CALL actores_por_pelicula(10);
117

```

**Result Grid**

nombre_actor
ALEC WAYNE
JUDY DEAN
VAL BOLGER
RAY JOHANSSON
RENEE TRACY
JADA RYDER

**Action Output**

#	Time	Action	Message	Duration / Fetch
28	07:36:09	CREATE PROCEDURE copias_por_pelicula(IN id_pelicula INT, OUT cantidad INT) BEG...	0 row(s) affected	0.031 sec
29	07:36:09	CALL copias_por_pelicula(10, @cantidad)	1 row(s) affected	0.000 sec
30	07:36:09	SELECT @cantidad LIMIT 0, 500	1 row(s) returned	0.000 sec / 0.000 sec
31	07:36:54	CREATE PROCEDURE actores_por_pelicula(IN id_pelicula INT) BEGIN SELECT CO...	0 row(s) affected	0.047 sec
32	07:36:54	CALL actores_por_pelicula(10)	8 row(s) returned	0.000 sec / 0.000 sec

Recibir el `customer_id` y devolver el correo electrónico del cliente.



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

```

144 • SELECT @total;
145
146
147
148 DELIMITER $
149 • CREATE PROCEDURE peliculas_por_clasificacion(IN clasificacion VARCHAR(10))
150 BEGIN
151     SELECT title, rating
152     FROM film
153     WHERE rating = clasificacion;
154 END $
155 DELIMITER ;
156 • CALL peliculas_por_clasificacion('PG');

```

Result Grid

title	rating
ACADEMY DINOSAUR	PG
AGENT TRUMAN	PG
ALASKA PHANTOM	PG
ALI FOREVER	PG
AMADEUS HOLY	PG
ARIZONA BANG	PG

Result 11

Output

Action Output

#	Time	Action	Message	Duration / Fetch
36	07:42:09	CREATE PROCEDURE contar_clientes_por_ciudad(IN ciudad VARCHAR(50), OUT tota...	0 row(s) affected	0.000 sec
37	07:42:09	CALL contar_clientes_por_ciudad('Athenai', @total)	1 row(s) affected	0.016 sec
38	07:42:09	SELECT @total LIMIT 0, 500	1 row(s) returned	0.000 sec / 0.000 sec
39	07:42:41	CREATE PROCEDURE peliculas_por_clasificacion(IN clasificacion VARCHAR(10)) BEG...	0 row(s) affected	0.031 sec
40	07:42:41	CALL peliculas_por_clasificacion('PG')	194 row(s) returned	0.016 sec / 0.000 sec

Recibir el estado (activo o inactivo) y mostrar la lista de clientes.

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

```

156 • CALL peliculas_por_clasificacion('PG');
157
158
159 DELIMITER $
160 • CREATE PROCEDURE clientes_por_estado(IN estado TINYINT)
161 BEGIN
162     SELECT customer_id, first_name, last_name
163     FROM customer
164     WHERE active = estado;
165 END $
166 DELIMITER ;
167 • CALL clientes_por_estado(1);
168

```

Result Grid

customer_id	first_name	last_name
1	MARY	SMITH
2	PATRICIA	JOHNSON
3	LINDA	WILLIAMS
4	BARBARA	JONES
5	ELIZABETH	BROWN
6	JENNIFER	DAVIS

Result 12

Output

Action Output

#	Time	Action	Message	Duration / Fetch
38	07:42:09	SELECT @total LIMIT 0, 500	1 row(s) returned	0.000 sec / 0.000 sec
39	07:42:41	CREATE PROCEDURE peliculas_por_clasificacion(IN clasificacion VARCHAR(10)) BEG...	0 row(s) affected	0.031 sec
40	07:42:41	CALL peliculas_por_clasificacion('PG')	194 row(s) returned	0.016 sec / 0.000 sec
41	07:46:54	CREATE PROCEDURE clientes_por_estado(IN estado TINYINT) BEGIN SELECT cu...	0 row(s) affected	0.000 sec
42	07:46:54	CALL clientes_por_estado(1)	504 row(s) returned	0.000 sec / 0.000 sec

Recibir un número y mostrar las películas con duración menor a ese valor.

Query 1 SQL File 5\* SQL File 5\* x

Limit to 500 rows

```

169
170 DELIMITER $
171 • CREATE PROCEDURE peliculas_por_duracion(IN duracion INT)
172 BEGIN
173     SELECT film_id, title, length
174     FROM film
175     WHERE length < duracion;

```

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

Result Grid

film_id	title	length
1	ACADEMY DINOSAUR	86
2	ACE GOLDFINGER	48
3	ADAPTATION HOLES	50
7	AIRPLANE SIERRA	62
8	AIRPORT POLLOCK	54
10	ALADDIN CALENDAR	63
15	ALIEN CENTER	46
17	ALONE TRIP	82
18	ALTER VICTORY	57
20	AMELIE HELLFIGHTERS	79
22	AMISTAD MIDSUMMER	85
25	ANGELS LIFE	74
26	ANNIE IDENTITY	86

Result 1 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	20:21:27	CREATE PROCEDURE peliculas_por_duracion(IN duracion INT) BEGIN SELECT film_i...	Error Code: 1046. No database selected Select the default DB to be used by double-clickin...	0.000 sec
2	20:21:33	USE sakila	0 row(s) affected	0.000 sec
3	20:21:35	CREATE PROCEDURE peliculas_por_duracion(IN duracion INT) BEGIN SELECT film_i...	0 row(s) affected	0.032 sec
4	20:21:36	CALL peliculas_por_duracion(90)	320 row(s) returned	0.047 sec / 0.000 sec

Recibir una fecha y listar los clientes registrados después de esa fecha.

Query 1 SQL File 5\* SQL File 5\* x

Limit to 500 rows

```

180 Execute the selected portion of the script or everything, if there is no selection
181 DELIMITER $
182 • CREATE PROCEDURE clientes_despues_fecha(IN fecha DATE)
183 BEGIN
184     SELECT customer_id, first_name, last_name, email, create_date
185     FROM customer
186     WHERE create_date > fecha;
187 END $
188 DELIMITER ;
189 • CALL clientes_despues_fecha('2006-01-01');
190
191
192

```

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

Result Grid

customer_id	first_name	last_name	email	create_date
1	MARY	SMITH	MARY.SMITH@sakilacustomer.org	2006-02-14 22:04:36
2	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	2006-02-14 22:04:36
3	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	2006-02-14 22:04:36
4	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org	2006-02-14 22:04:36
5	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacustomer.org	2006-02-14 22:04:36

Result 2 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
2	20:21:33	USE sakila	0 row(s) affected	0.000 sec
3	20:21:35	CREATE PROCEDURE peliculas_por_duracion(IN duracion INT) BEGIN SELECT film...	0 row(s) affected	0.032 sec
4	20:21:36	CALL peliculas_por_duracion(90)	320 row(s) returned	0.047 sec / 0.000 sec
5	20:22:06	CREATE PROCEDURE clientes_despues_fecha(IN fecha DATE) BEGIN SELECT cu...	0 row(s) affected	0.000 sec
6	20:22:06	CALL clientes_despues_fecha('2006-01-01')	599 row(s) returned	0.000 sec / 0.000 sec

Recibir un store\_id y mostrar todas las películas disponibles en esa tienda.

Query 1 SQL File 5\* SQL File 5\*

Limit to 500 rows

Execute the selected portion of the script or everything, if there is no selection

```

192 CREATE PROCEDURE peliculas_en_tienda(IN tienda_id TINYINT)
193 BEGIN
194     SELECT DISTINCT f.film_id, f.title
195     FROM film f
196     JOIN inventory i ON f.film_id = i.film_id
197     WHERE i.store_id = tienda_id;
198 END $
199 DELIMITER ;
200 CALL peliculas_en_tienda(1);
201
202
203
204

```

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

Result Grid

film_id	title
1	ACADEMY DINOSAUR
4	AFFAIR PREJUDICE
6	AGENT TRUMAN
7	AIRPLANE SIERRA
9	ALABAMA DEVIL

Result 3

Output

Action Output

#	Time	Action	Message	Duration / Fetch
4	20:21:36	CALL peliculas_por_duracion(90)	320 row(s) returned	0.047 sec / 0.000 sec
5	20:22:06	CREATE PROCEDURE clientes_despues_fecha(IN fecha DATE) BEGIN SELECT cu...	0 row(s) affected	0.000 sec
6	20:22:06	CALL clientes_despues_fecha('2006-01-01')	599 row(s) returned	0.000 sec / 0.000 sec
7	20:22:25	CREATE PROCEDURE peliculas_en_tienda(IN tienda_id TINYINT) BEGIN SELECT ...	0 row(s) affected	0.031 sec
8	20:22:25	CALL peliculas_en_tienda(1)	759 row(s) returned	0.000 sec / 0.000 sec

TODO EL CODIGO:

USE sakila;

DROP PROCEDURE IF EXISTS obtener\_nombre\_completo

DELIMITER \$

CREATE PROCEDURE obtener\_nombre\_completo(IN id\_cliente INT, OUT  
nombre\_completo VARCHAR(100))

BEGIN

SELECT CONCAT(first\_name, ' ', last\_name)

INTO nombre\_completo

FROM customer

WHERE customer\_id = id\_cliente;

END \$

DELIMITER ;

CALL obtener\_nombre\_completo(1, @nombre);

SELECT @nombre;



**DROP PROCEDURE IF EXISTS contar\_peliculas\_por\_categoria**

**DELIMITER \$**

**CREATE PROCEDURE contar\_peliculas\_por\_categoria(IN nombre\_categoria  
VARCHAR(50), OUT total INT)**

**BEGIN**

**SELECT COUNT(\*)**

**INTO total**

**FROM film\_category fc**

**JOIN category c ON fc.category\_id = c.category\_id**

**WHERE c.name = nombre\_categoria;**

**END \$**

**DELIMITER ;**

**CALL contar\_peliculas\_por\_categoria('Action', @total);**

**SELECT @total;**

**DELETE PROCEDURE IF EXISTS existe\_cliente**

**DELIMITER \$**

**CREATE PROCEDURE existe\_cliente(IN id\_cliente INT, OUT existe TINYINT)**

**BEGIN**

**SELECT COUNT(\*) > 0**

**INTO existe**

**FROM customer**

**WHERE customer\_id = id\_cliente;**

**END \$**

**DELIMITER ;**

**CALL existe\_cliente(1, @existe);**

**SELECT @existe;**

**DELIMITER \$**

**CREATE PROCEDURE peliculas\_mayor\_duracion(IN min\_duracion INT)**

**BEGIN**

**SELECT title, length**

**FROM film**

**WHERE length > min\_duracion;**

**END \$**

**DELIMITER ;**

**CALL peliculas\_mayor\_duracion(120);**

**DELIMITER \$**

**CREATE PROCEDURE peliculas\_rentadas\_por\_cliente(IN id\_cliente INT)**

**BEGIN**

**SELECT DISTINCT f.title**

**FROM rental r**

**JOIN inventory i ON r.inventory\_id = i.inventory\_id**

**JOIN film f ON i.film\_id = f.film\_id**

**WHERE r.customer\_id = id\_cliente;**

**END \$**

**DELIMITER ;**

**CALL peliculas\_rentadas\_por\_cliente(1);**

**DELIMITER \$**

**CREATE PROCEDURE total\_dinero\_por\_tienda(IN id\_tienda INT, OUT total  
DECIMAL(10,2))**

**BEGIN**

**SELECT SUM(p.amount)**

**INTO total**

**FROM payment p**

**JOIN rental r ON p.rental\_id = r.rental\_id**

**JOIN inventory i ON r.inventory\_id = i.inventory\_id**

**WHERE i.store\_id = id\_tienda;**

**END \$**

**DELIMITER ;**

**CALL total\_dinero\_por\_tienda(1, @total);**

**SELECT @total;**

**DELIMITER \$**

**CREATE PROCEDURE copias\_por\_pelicula(IN id\_pelicula INT, OUT cantidad INT)**

**BEGIN**

**SELECT COUNT(\*)**

**INTO cantidad**

**FROM inventory**

**WHERE film\_id = id\_pelicula;**

**END \$**

**DELIMITER ;**

**CALL copias\_por\_pelicula(10, @cantidad);**

**SELECT @cantidad;**

**DELIMITER \$**

**CREATE PROCEDURE actores\_por\_pelicula(IN id\_pelicula INT)**

**BEGIN**

**SELECT CONCAT(a.first\_name, ' ', a.last\_name) AS nombre\_actor**

**FROM film\_actor fa**

**JOIN actor a ON fa.actor\_id = a.actor\_id**

**WHERE fa.film\_id = id\_pelicula;**

**END \$**

**DELIMITER ;**

**CALL actores\_por\_pelicula(10);**

**DROP PROCEDURE IF EXISTS correo\_cliente**

**DELIMITER \$**

```
CREATE PROCEDURE correo_por_cliente(IN id_cliente INT, OUT correo VARCHAR(100))
BEGIN
    SELECT email
    INTO correo
    FROM customer
    WHERE customer_id = id_cliente;
END $
DELIMITER ;
CALL correo_por_cliente(1, @correo);
SELECT @correo;
```

DELIMITER \$

```
CREATE PROCEDURE contar_clientes_por_ciudad(IN ciudad VARCHAR(50), OUT total
INT)
BEGIN
    SELECT COUNT(*)
    INTO total
    FROM customer cu
    JOIN address a ON cu.address_id = a.address_id
    JOIN city c ON a.city_id = c.city_id
    WHERE c.city = ciudad;
END $
DELIMITER ;
CALL contar_clientes_por_ciudad('Athenai', @total);
SELECT @total;
```

DELIMITER \$

```
CREATE PROCEDURE peliculas_por_clasificacion(IN clasificacion VARCHAR(10))
BEGIN
    SELECT title, rating
    FROM film
    WHERE rating = clasificacion;
```

**END \$**

**DELIMITER ;**

**CALL peliculas\_por\_clasificacion('PG');**

**DELIMITER \$**

**CREATE PROCEDURE clientes\_por\_estado(IN estado TINYINT)**

**BEGIN**

**SELECT customer\_id, first\_name, last\_name**

**FROM customer**

**WHERE active = estado;**

**END \$**

**DELIMITER ;**

**CALL clientes\_por\_estado(1);**

**DELIMITER \$**

**CREATE PROCEDURE peliculas\_por\_duracion(IN duracion INT)**

**BEGIN**

**SELECT film\_id, title, length**

**FROM film**

**WHERE length < duracion;**

**END \$**

**DELIMITER ;**

**CALL peliculas\_por\_duracion(90);**

**DELIMITER \$**

**CREATE PROCEDURE clientes\_despues\_fecha(IN fecha DATE)**

**BEGIN**

**SELECT customer\_id, first\_name, last\_name, email, create\_date**

**FROM customer**

**WHERE create\_date > fecha;**

**END \$**

**DELIMITER ;**

**CALL clientes\_despues\_fecha('2006-01-01');**

**DELIMITER \$**

**CREATE PROCEDURE peliculas\_en\_tienda(IN tienda\_id TINYINT)**

**BEGIN**

**SELECT DISTINCT f.film\_id, f.title**

**FROM film f**

**JOIN inventory i ON f.film\_id = i.film\_id**

**WHERE i.store\_id = tienda\_id;**

**END \$**

**DELIMITER ;**

**CALL peliculas\_en\_tienda(1);**